Issue 31 | Dec 2010

Blender learning made easy

blenderart magazine

Under the Microscope

Physics of Circular Motion

A World of Rotations

BioBlender: Blender for Biologists

Computer Simulation and Modeling of Liquid Droplets

COVERART Virus -by Adam Auksel

EDITOR
Gaurav Nawani gaurav@blenderart.org

MANAGING EDITOR
Sandra Gilbert sandra@blenderart.org

WEBSITE
Nam Pham nam@blenderart.org

DESIGNER/LAYOUTING
Gaurav Nawani/Nikhil Rawat

PROOFERS
Brian C. Treacy
Bruce Westfall
Daniel Hand
Daniel Mate
Henriël Veldtmann
Joshua Leung
Joshua Scotton
Kevin Braun
Mark Warren
Noah Summers
Patrick ODonnell
Phillip
Ronan Posnic
Scott Hill
Wade Bick
Valérie Hambert

WRITERS
Robert T
Enrique_Sahagun
di mistic
Pep Ribal
Francisco M. Gomez-Campos
Jamie Tica
Richard Charvát
Adam Kalisz
Enrique_Sahagun
Raluca Mihaela Andrei
Mike Pan
Monica Zoppè
Rod Cockcroft

COVER ART
Virus -by Adam Auksel

CONTENTS

Sandra Gilbert
Managing Editor

"Community support is a huge draw for many new users, and in my opinion, one of the greatest advantages to choosing Blender."

The greater majority of Blender users focus on the modeling and animation aspects of Blender. And with good reason. Blender has steadily grown and improved its toolset over the years, giving both hobbyists and professionals alike a powerful 3D program.

Not surprisingly, Blender's growth has also drawn a new user base from the scientific community. This year alone the annual Blender Conference saw a number of exciting scientific/educational presentations. Several factors make Blender ideal for scientific projects.

As much as we would all like to pretend otherwise, cost is a factor. Since Blender is free, precious research funds can remain focused on the research itself.

Blender provides an excellent tool set for producing simulations, visualizations and walk-throughs, as well as educational videos based on the various research projects.

The open nature of Blender, as well as the ease of creating python add-ons allows for complete customization of Blender to the needs of the project (e.g. Bio-Blender).

Community support is a huge draw for many new users, and in my opinion, one of the greatest advantages to choosing Blender. The Blender community has always excelled at helping new and existing users solve problems.

Fast development also makes Blender an attractive option. It is not an uncommon occurrence to notice a bug/problem and seemingly overnight the solution has been coded and uploaded. Many times, I no sooner think, "Gee, "X" feature would be so helpful". Then I go to research if there is a work around, only to discover that someone has anticipated my wish and it is already coded. Now that is "CUSTOMER SERVICE".

As you might have guessed, in this issue we delve "Under the Microscope" as we explore some of the exciting ways Blender is being used to further education and research. So grab a hot beverage of choice and settle in for an enlightening experience ■

"After learning how to create your own lovely new artworks, you might want to kick it up a notch and get a little motion going."

I have always been fascinated by electron microscope style images. They often display a fragile almost magical quality that is just beautiful. Over the years, blender users have come up with a number of creative methods for producing these lovely images.

Blender 2.5 of course, has brought new tools and techniques to the election microscope look. Here's a couple of tutorials that I have run across just in the last few months by very talented Blender users that make excellent use of Blender 2.5 to created truely beautiful "Electron Microscope" style images.

Creating a Microscopic Virus Effect

Blendercookie.com has become a major educational resource since it's launch. Among the numerous video tutorials covering a dizzying array of topics, there is a beautiful tutorial by Jonathan Williamson on creating a Microscopic Virus.

In addition to showing how to quickly and easily model a "virus", he shows the viewer how to achieve that lovely electron microscope look. The end result is beautiful and easy enough for even a beginner to accomplish in little to no time.

Construct a 'Microcosm' Using Blender 2.5

This is a 4 part in depth video tutorial series by ***Frederik Steinmetz*** on how to create a Microcosm. Each video covers a different part of the process, leading to an absolutely beautiful end result.

From cg.tuts.com

In the first part of this brand-new, advanced Blender 2.5 tutorial, *Frederik Steinmetz* walks us through how he modeled and rigged his 'Microcosm' creation! The full tutorial will go on to discuss complex particle & hair simulation, how to add materials, texture and light the scene, and finally how to use nodes for depth of field.

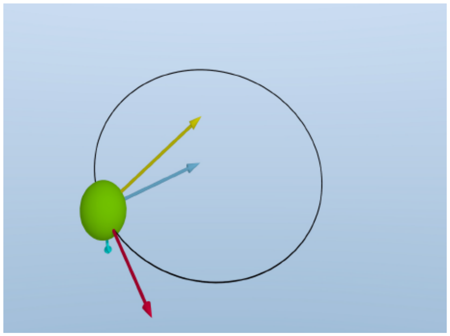
In the second part of his brand-new, advanced Blender 2.5 tutorial, *Frederik Steinmetz* continues to walk us through how he created his 'Microcosm' scene! After modeling the scene in part 1, today's tutorial deals with particle and hair simulation, whilst later parts go on to cover adding materials, how to texture and light the scene, and finally how to use nodes for depth of field.

In the third part of his brand-new, advanced Blender 2.5 tutorial, Frederik Steinmetz continues to walk us through how he created his 'Microcosm' scene!

After modeling the scene in part 1, and dealing with the particle and hair simulations in part two, today's part starts looking at how to texture and light the scene!

In the forth and final part of his brand-new, advanced Blender 2.5 tutorial, *Frederik Steinmetz* continues to walk us through how he created his 'Microcosm' scene! After modeling the scene in part 1, dealing with the particle and hair simulations in part two and texturing and lighting the scene in part three, today it's all about finalising the scene, rendering and putting together the final composite! It's time to wrap up this awesome series.

After learning how to create your own lovely new artworks, you might want to kick it up a notch and get a little motion going. Recently ***Dimistic*** sent me a fun little blend file for you to play with, that shows his method for creating an electron in motion, as well as the settings he used to create a rather nice glow/glare effect. The blend file is included in this issue's blend zip download. You can see a short animation of his electron motion on youtube



by- Adam Kalisz

Introduction

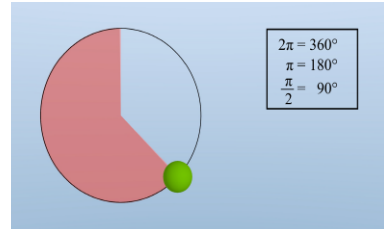
The first moment I heard about the new topic for the current #31 BlenderArt Magazine, I was like 'Cool, I have to participate!' Unfortunately I didn't know how to contribute something. But since currently I'm attending a vocational school in Nuremberg, Germany, finding a suitable topic was not such a big deal. Furthermore, I think that I will deeply benefit from this. On the one hand I can improve my physics skills and on the other hand my English skills. Ah, and of course my Blender skills as well! ;)

Approaching the animation

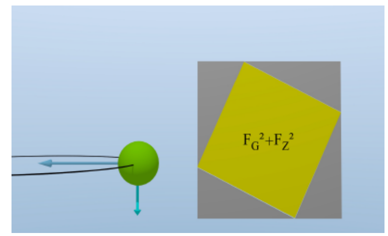
So I sat down and thought about my animation concept. Since we were dealing with circular motion in physics and I knew that my classmates and I had several problems with applying the corresponding laws, I decided to stick to that topic. With the formulas lying in front of me, I started to write the script. I had to think about a didactical (teaching) structure and at the same time how to visualize it in Blender.

In the end, the speaker text took up a whole page and after the first audio recording I was able to start the realisation in Blender. The cool thing with these projects is that Blender has an internal video sequencer, where you can just put all of your audio files and play them back in real time. Because of that, there weren't any synchronisation problems and the animation process was very comfortable.

I wanted to keep everything as dynamic as possible, so I looked for constraints to help with the animation. Finally I used a 'Follow Path' constraint and took the opportunity to keyframe the offset parameter. To scale the vectors against the radius of the circle, the 'Copy Scale' constraint did a great job. But I had to set keyframes for the influence to stay put at 0 until the respective vectors are mentioned, because otherwise they would scale up too early, e.g. when scaling up the radius of the circle when the radial acceleration has not been mentioned yet.



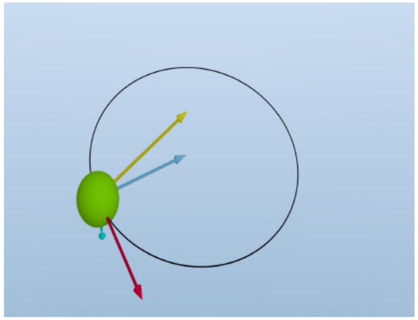
The rest was made by creating keyframes to animate the vectors and using shape keys for the Pythagorean theorem. It turned out that I had forgotten to change the frame rate to the European standard of 25 frames per second. Because of that I had to readjust all of the keyframes. That's why it's essential you set up your render settings before you begin animating. This brings us to the problems I had to face in the animation process.



Problems

Fortunately, since Blender can save the sound into the final video due to its sequencer, there was only one big problem: The text animation.

I decided to do it in After Effects, which turned out to be a very time-consuming process.

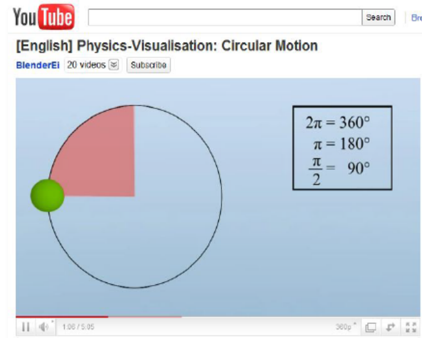


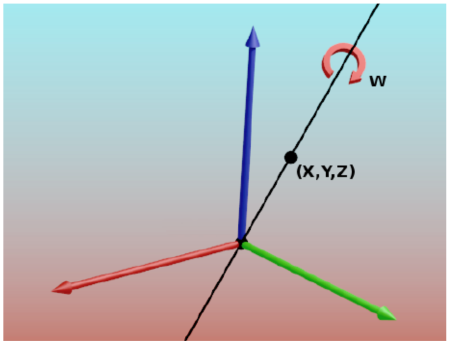
After Effects doesn't support the audio playback in real time, at least not until CS3. So I had to switch on the display of seconds in the timeline in Blender and tweak all of the fade in and fade out animations in After Effects synchronously while scrubbing through the animation in Blender. Hopefully in future, Blender will get a sophisticated tool to add some text in the compositor to avoid third party software, which brings more trouble than it actually helps, at least as far as synchronisation is concerned.

Conclusion

To conclude, the development of this video was a rather simple task. Blender was a great aid with its built-in animation features allowing for a dynamic, customisable animation process. I really appreciate Blender's big arsenal of tools and the organised graphical user interface. It's an indispensable Open Source graphics suite that everybody should use and support. I'm very enthusiastic about it and founded the first Blender user group in Nuremberg with some other guys and I'm trying to establish a quality German video tutorial website to increase the number of Blender users in Germany. Thanks to the Blender Foundation and all of the developers!!!

You can watch the animation [here](): ■





by- Pep Ribal

Introduction

What is it with rotations that makes them so frightening?

Actually rotations are very useful, and sometimes absolutely necessary. Imagine a world without rotations... Tire manufacturers will agree with me... Indeed, of the three kinds of transformations (translation, rotation and scaling), rotations are by far the most complex. Let's see why.

Take the Blender default scene. Activate the Transform panel (hotkey **N**). Make sure the Translate manipulator is on, and Transform Orientation is set to Global. Now move the default cube along the X axis using the manipulator (red arrow). Take a look at the Transform panel as you drag the cube. You'll see the Location X value change as you do, while the other values remain unchanged. OK, drop the cube where you wish. Now do the same along the Y axis, and you'll see Location Y change as you drag. Once more, the rest of the values remain unchanged. Finally, you can see the same thing happens with Z axis.

Then you can change the 3D manipulator to Scale. Keep trying with the three axes, and you'll realize that each modification affects only its own axis value (Scale values). It also changes the Dimensions values, but these are not relevant, as they refer to the final dimensions of the mesh, not to the transform properties of the object.

Rotating an object

First of all, a brief description of the Transform Orientations available for the 3D manipulators in Blender. View has a set of axes aligned with the viewport direction, Normal is aligned with the normal of the actual object data selection (like mesh faces) in Edit Mode, and it's equivalent to Local orientation in Object Mode, Local is aligned with the object local coordinate system, and Global is aligned to the world coordinate system. We will see later what Gimbal means.

Once that's said, let's start the show. First make sure XYZ Euler is selected in the Transform panel. Try now with the Rotate manipulator, with Global orientation. Drag around the Z axis (blue ring). You can also use hotkey R, and then Z for rotation around the Z global axis. You can see the Rotation Z value change as you rotate. Drop it at will. Now rotate around any of the other two axes... What happens? All three rotation values (X, Y and Z) change as you drag...

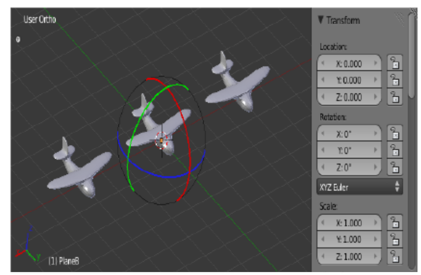


Figure 1. Initial state

We have just discovered that rotation around one axis affects the value of the other two. Let's go deeper into this. Open the provided file 'RotationsWorld.blend'. There you have three simple airplanes (fig 1).

We will use the Rotate manipulator to perform three rotations on them: 120º around the X global axis, 60º around Y, and 45º around Z. But we will change the order of those rotations in each object.

Bear in mind that positive angles mean counterclockwise rotations.

Start with 'PlaneA'. Use the manipulator to rotate X axis first. Check the amount of rotation in the 3D view header, not in the Transform panel. Use the **Ctrl** key to round up the rotated value to 120º as you drag. If you use hotkeys **R** and then **X**, you can also enter 120 with the keyboard. Next, rotate 60º around Y axis and finally 45º around Z axis.

Now perform the same rotations on 'PlaneB' but in this order: first 60º around Y, then 120º around X, and last, 45º around Z. When done, go for 'Plane C', using a new order: 45º around Z, 120º around X, and 60º around Y (figure 2). Remember to always check the rotation amount in the 3D view header only.

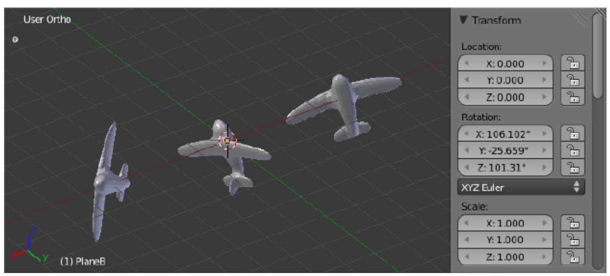


Figure 2. After rotations around global axes.

OK, what do we have now? Three airplanes with a completely different orientation in space. If you take a look at the rotation values of the three planes, only 'PlaneA' keeps the values of the applied rotation (X=120º, Y=60º, Z=45º), while the others hold very strange numbers. You can see that the order of rotation is important. Even if we use Local mode for rotation manipulators, the problem doesn't improve (figure 3). For rotation

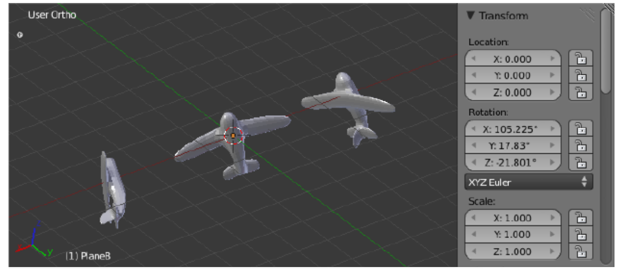


Figure 3. After rotations around local axes.

around X local axis for instance, you can also press **R, X, X.**

In translation and scaling we can just enter the values we wish into the Transform panel manually, as there is only one way to interpret their meanings. But as we have just seen, with rotations, entering the values X=120º, Y=60º, Z=45º in the slider controls might not yield the desired result. If we were looking for the orientation of 'PlaneA', that would have been OK. But if we wanted for instance, the rotation of any of the other two, that wouldn't have done the trick.

We need a rotation system with a special set of axes that lets us forget about the order, so that we can type the three rotation angles directly in the Transform panel, or use a manipulator so that each ring affects only one axis value. And that's exactly what Blender does. It's not using the global or local axes, as you have seen by the strange numbers you got in the rotation values of the objects. So what is that wonderful system that Blender uses internally?

Euler rotations

We have previously set rotation mode to XYZ Euler. That is exactly what Blender is using internally. The best way to see these type of rotations in action is to set the Transform Orientation of the 3D manipulator to Gimbal. This widget lets you see the current state of the Euler rotation transform.

A gimbal is a circular gadget that spins around an axis that goes through one of its diameters. If you mount three of these one inside the other, you have a 3-axes gimbal (figure 4). This kind of device is used in gyroscopes, for instance.

The Blender Gimbal rotation manipulator closely resembles one of these gadgets, as in a 3-axes gimbal these move in relation to the others. However, Blender gimbal is a bit different from that, the main difference being the axis of rotation of the rings as you can see in figures 5 and 6. While the physical gimbal rotates around one of its diameters (figure 5), each ring of the Blender gimbal rotates around an axis that goes through the centre of the ring and is perpendicular to all of its diameters (figure 6).

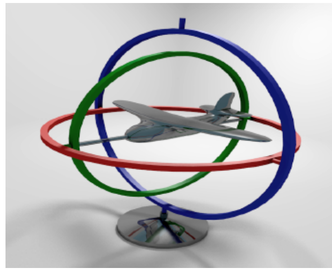


Figure 4. A gimbal gadget.

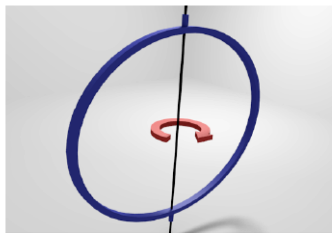


Figure 5. Physical gimbal rotation.

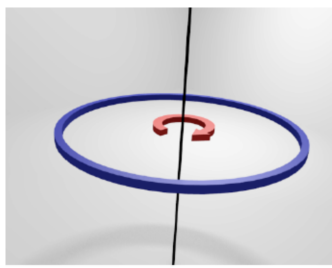


Figure 6. Blender gimbal rotation.

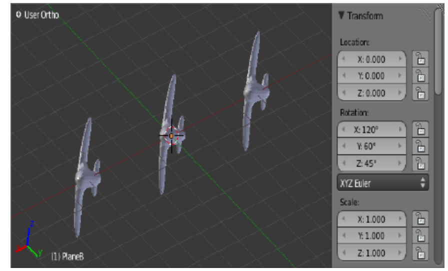


Figure 7. After Euler rotations.

So, let's start playing with the gimbal. Take any object with 0 rotation. Now activate the Gimbal manipulator. Set the rotation mode to XYZ Euler (though it would work with any other Euler type). And now start rotating the axes individually. You can repeat the experiment of the three airplanes, and you'll get the results of figure 7. See what happens in the Transform panel.

Now, each gimbal axis is directly related to the corresponding rotation value of the object. What does it mean? That order of rotation doesn't matter. Maybe you have realized that all three airplanes end up in the same position using the Euler gimbal. If so, you will have seen that all three airplanes have the same rotation values in the Transform panel. In other words, you can enter the desired rotation angles numerically in the slider controls.

So, what's the difference between a local or global rotation system and the gimbal system? And why are there six different types of Eulers? And why am I asking all this if I know the answer...?

As you can see in a physical 3-axis gimbal gadget, there are three axes configured in such a way that they form a hierarchy. When we rotate the outermost ring, the one on top of the hierarchy, we are actually rotating the entire system around the axis of that ring.

Rotating the middle ring, we can see the innermost ring rotate as well. If we rotate the innermost ring, only that ring moves.

With Blender gimbals the same thing happens. So we must choose one axis to be on top of the hierarchy, one to be in the middle, and the last one to be at the bottom.

Let's say we want the Z axis to be on top; the X axis to be its child; and finally Y to be at the bottom. In other words, Z axis will be the parent of X, and X the parent of Y. In order from bottom to top, we have Y, X, and Z. That forms a YXZ gimbal, used for YXZ Euler rotations.

There are six different combinations of hierarchies with the three axis X, Y and Z, and therefore, six different kinds of gimbals, each one of which is associated to its corresponding Euler rotation system. With Eulers, it's important to remember that the axis written first is the one at the bottom of the hierarchy, while the last one on the right is the one on top of it. Thus, in a XYZ Euler, the Z axis in on top, while X is at the bottom.

Blender uses two things to calculate the Euler rotation of an object. First, the values of the three rotations around each of the three axes (X,Y and Z); and second, what type of Euler hierarchy are these values based on. For instance it's not the same to use a XYZ or a ZXY hierarchy. You can check this by taking the rotated airplane. Don't change the rotation values, just change to any of the other five Euler modes. You will immediately see the final rotation changes.

When Blender has calculated the rotation of the object (using the Eulers), it stores that rotation in the object matrix, which is basically a 4x4 matrix of numbers that keep track of the full transformation state of the object: its location, rotation and size. When you are just modelling (not animating), it doesn't make any difference which rotation mode you are using, as they will all end up in the same place internally, i.e. the matrix. No use will be made of the Euler values. However when animating, Blender actually uses those Euler values to interpolate rotations, as we will see later.

Any of the Euler types has the advantages of isolating the effect of each axis, though they yield different rotations. Not a big deal. It's just a question of experimenting with them, and see how each type of gimbal behaves. OK, now we have found a magical rotation system that will make this world a better place for you and me... So, why do we need other rotation systems?

Euler rotation problems

If we want to define any orientation, or we want to rotate a face or a group of vertices, we can get them to rotate wherever we want to use any of the Euler modes. But when it comes to animation, we can run into some trouble in certain circumstances.When you want to animate a rotating object you have to use the same system from one keyframe to the other. You cannot start defining a XYZ Euler orientation for one keyframe, and then a YZX Euler for the next one. Why?

Because Blender interpolates between two rotations using the values of the specific system used (Euler or any other); it doesn't use the rotation stored in the object matrix. So if you use a different system in two consecutive keyframes, there is no way to calculate the interpolation values between them.

Let's go for another experiment. Open the file 'RotationsWorld.blend'. Make sure that the airplanes don't have any rotation applied, then go to frame 1 and select ZXY Euler rotation mode. We will focus on one of the airplanes as we are going to make it do some aerobatics. You can delete the other two if you want.

The first thing to do will be to set the Transform Orientation to Local, so that we can manipulate our airplane easily. Bear in mind however, that even if the manipulator is set to Local, Blender is using ZXY Euler internally to compute rotations and to interpolateangles, so what Blender is actually using is the ZXY gimbal. Now insert a rotation keyframe in frame 1, with the airplane in rest position (no rotations at all) as shown in figure 8.

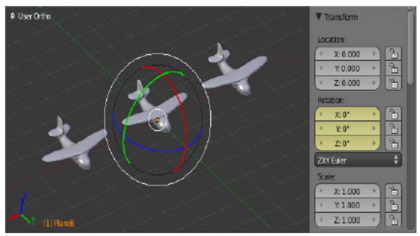


Figure 8. Keyframe 1 (local axes shown).

Now let's move to frame 25 using the arrow keys. In this frame, the pilot has astonished the audience of the airshow by setting the airplane in vertical position. Rotate 90º around the X local axis (remember that positive angles mean counterclockwise rotations), and set a new rotation keyframe (figure 9). Now the nose of the plane is pointing up. You can switch from Local to Global mode to see how the local axis has change. The "up" side of the plane is not the same as the "up" side of the world.

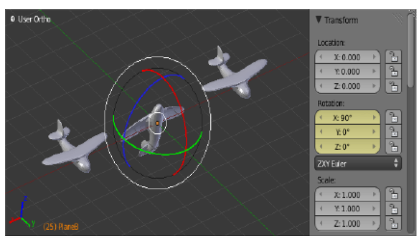


Figure 9. Keyframe 2 (local axes shown).

OK. But the pilot, who is a really bold guy, hasn't had enough. He wants to make a nice turn to his right while keeping the aircraft nose up. So now, get back to Local mode and go to frame 50. Then use the manipulator to rotate the airplane 90º around the Y local axis. Set a new rotation keyframe (figure 10).

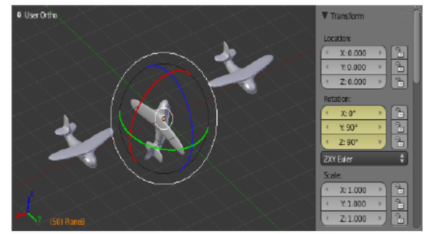


Figure 10. Keyframe 3 (local axes shown).

Now rewind to frame 1, and check the full animation using the arrow keys back and forth. You'll see that from keyframe 1 (frame 1) to keyframe 2 (frame 25) everything works as expected. But something weird happens between keyframe 2 and keyframe 3 (frame 50). We expect a right turn, but the airplane nose actually makes a weird movement.

To check what the problem was, set Gimbal orientation, rewind to frame 1, and check the animation again. As you approach frame 25, the Z rotation axis of the manipulator gets closer and closer to Y axis. At frame 25, the Z axis is completely aligned with the Y axis, as shown in figure 11.

We have just lost one axis of movement. This phenomenon is known as the Gimbal lock, and it's a very typical source of headache for animators.

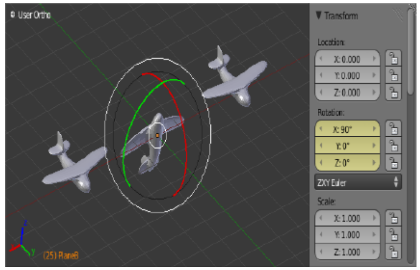


Figure 11. Keyframe 2 (gimbal shown)
Where is the Z axis...? Perfect gimbal lock

You may have found that your animated rotations behave in a weird manner (it has indeed happened to me), and there is no way to fix them no matter what you do to avoid the problem. Well, it's more than likely that you have been a victim of the hideous gimbal lock (ock... ock... ock...).

So, back to our airplane. We are nose up, and we have lost an axis to perform a right turn. For doing that kind of turn, we would need to get our Z axis back. Well, if you check the interpolation values of the animation between keys 2 and 3, you will realize that this is exactly what Blender does. While the Y axis rotates the commanded 90º, it also undoes the initial 90º rotation in the X axis (which caused the gimbal lock), and rotates 90º around the Z axis so that the final position can be reached.

The end result is the weird movement of the airplane. Unfortunately, that caused the audience to go back home, and the airshow resulted in a complete failure. Let's try to see when gimbal lock occurs, taking into account the type of Euler rotation we choose, so that we can avoid it.

We know there are three rotation axes in a gimbal. When all three axes are perpendicular between them, all is fine. However, as one of the axes starts to move towards another, they lose their relative perpendicularity, meaning that we are starting to lose some degree of freedom of movement. The problem reaches its maximum when two axes become completely aligned (parallel), that is, when we completely lose one of the three axes.

Let's take for instance a XYZ Euler gimbal. What happens when the axis at the bottom of the hierarchy (in this case, X) rotates? Nothing important actually. All three axes keep perpendicular whatever rotation you apply to the X axis, which just keeps spinning around itself.

What if we rotate the topmost axis in the hierarchy (Z)? Then all the axes in the system rotate with it, keeping their relative positions, without losing freedom of movement like before. The problem comes when we rotate the axis in the middle (Y). Its effect is to get its child axis (X) closer to its parent axis (Z). That said, one important thing to remember is that the middle axis in the Euler hierarchy is crucial, and we need to keep an eye on it most of all.

Now that we know when gimbal lock is reached, we can see how to avoid it. So, if you need an object to perform an animation with a series of rotations in which its Z axis will reach angles close to 90º (or equivalent angles like -90º or 270º), we will avoid the use of Euler rotation systems XZY and YZX, as in these, Z axis lays in the middle of the hierarchy.

However, we could still use XZY Euler even if the Z axis reaches 90º, but only if in those particular moments we don't need the X axis to rotate. We need to make sure that as soon as we need rotations around the X axis, Z rotation is far from 90º (and equivalents).

If you want to perform the former aerobatics, you can choose a different Euler system. For instance, you can repeat the experiment with a XYZ Euler system, and you will see everything working fine.

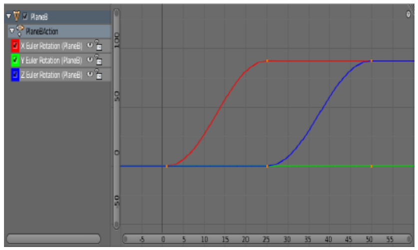


Figure 12. Euler rotations: animation curves.

When you are done with it, you can take a look at the resulting animation curves in the Graph editor (figure 12). See how intuitive those f-curves are. You can see a 90º rotation around the X axis take place between frame 1 and 25, and another 90º rotation around the Z axis between frames 25 and 50.

This is one of the big advantages of the Euler rotations as you can directly manipulate the rotation f-curves easily, knowing that the curves are independent between them. Those three curves give you a clear picture of what's going on, even if you don't see the actual object.

In this case all you have to do is keep an eye on the green curve (Y axis) and make sure it doesn't approach 90º when the red curve (X axis) is different than zero.

OK, but is there any rotation system which doesn't suffer from gimbal lock...?

Sure there is.

Axis Angle rotations

If you set the rotation mode to Axis Angle, you will notice that you now have 4 values for defining rotations: X, Y, Z and W.

With Euler we had 3 values representing a rotation angle around each axis. With axis-angle we define two things: one axis and one angle. The axis is defined by X, Y and Z; the rotation angle, by W. You can see that in figure 13.

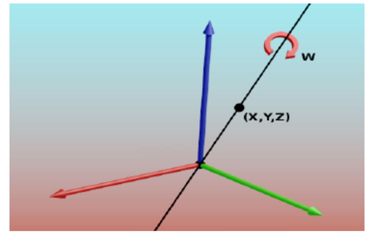


Figure 13. Axis-angle rotation.

The effective rotation is done around the axis (X,Y,Z). This axis is an infinite line that goes through the centre of the object and the point defined by (X,Y,Z) in the local coordinate system of the object. There are many ways to define the same axis. The most important thing is the ratio between these three values. Thus, (1,0.5,3) is the same axis as (2,1,6).

So once we have this rotation axis, all we have to do is to make the object rotate around it by the amount given in the W value. So if W=0, no rotation is applied, regardless of the values in X, Y and Z. Conversely, if X, Y and Z equal 0, no axis is defined, so once more, there will be no rotation regardless of the W value.

You can easily see that the most obvious advantage of axis-angle is rotation around an arbitrary axis. This makes axis-angle very suitable for objects that spin constantly around the same axis. The rotation of Earth around its peculiar axis is a perfect example.

Bear in mind that negative values matter in the axis definition, as axes have a direction. Two axes defined with opposite direction (simply changing the signs of X, Y, and Z) will have an opposite direction, and so, an inverse rotation. So changing the sign of all four values (axis and angle) has no effect on the final orientation (except for animation purposes).

One thing to note is that angle values in W are expressed in radians, not degrees. In that case, if you want to perform a 90º turn, you must rotate Π/2 in the W slider control. It's possible to type pi directly in the sliders as Blender knows its value (around 3.141592). You just need to know that 360 degrees are equivalent to 2xΠ radians, so that you can calculate other angles. You can directly enter values like pi/2, 3*pi/2, 2*pi, pi, etc. Anyway, manipulators and hotkey R always use degrees.

OK, now that we know what axis-angle is, it's time to play with it.

You can repeat the aerobatics experiment from scratch. Set rotation mode to Axis Angle, and redo the three keyframes using the 3D manipulator of choice, or hotkey **R**. Play the animation back.

What happens? Nothing good really.

Axis-angle rotation problems

As mentioned before, axis-angle works well for rotations around a fixed axis. So our aerobatics is not the best example to use. Let's see why it didn't work out smoothly (by now the pilot is depressed and already thinking about retiring).

What happened here is that from keyframe 1 to keyframe 2, two things were interpolated. First we went from axis (0,1,0) to axis (1,0,0). This axis movement is quite big, as it's going from one line to a perpendicular one (90º away). And second, we went from angle 0 to angle Π/2 (90º). So two things were moving the same amount: the axis and the angle.

Once again, let's go back to frame 1. Instead of axis (0,1,0), let's enter (1,0,0). It makes no difference, as the rotation angle value W is 0. Now update the keyframe and see how it goes.

Everything runs quite fine now. The second half is not perfect, but quite acceptable. Why isn't it perfect? It's important that between two consecutive keyframes most of the movement is taken by only one of the components: either the axis or the angle. In our initial airplane movement, both components were moving the same amount (90º), and that created a turbulent movement that made the pilot sick. Then we completely fixed the problem by keeping the axis still between keyframes. In the second half of the animation, the angle moves more than the axis, which is good, but both of them move.

If you want absolutely perfect movements, just move only one of the two components between two consecutive keyframes (usually the angle). Sometimes this is difficult, so the best thing in those situations is to start considering any other rotation system.

In axis-angle, rotation manipulators have to be used with special care, as they can lead to unwanted results. A simple rotation using the manipulator can lead, for instance, to the flipping (sign change) of the axis. If the initial axis is (0,1,0) and the final one results in (0,-1,0), that will produce, most probably, undesired effects, as we are changing its direction, which means a 180º rotation of the axis (not around the axis).

Moreover, in axis-angle you can define axes and angles using any value, as big or small as you want. Rotations can consist of several spins around the axis, that is, Nx2Π, where N is any number of revolutions, positive or negative. However, when using rotation manipulators you will always get axis values up to 1.0, and angles up to 2Π.

This, along with the possible axis flip, are good reasons to prefer editing rotation values directly on the Transform panel over using rotation manipulators or hotkey R.

To summarize, axis-angle is good for rotations around an arbitrary axis, as long as that axis doesn't keep moving, or at least its movement is really controlled. Remember that you can quickly move the axis to any value when rotation angle is 0 so that moment can be used to switch from one axis to another.

Now that you are done with the new air-show animation using axis-angle, take a look at the resulting animation curves (figure 14)... What can you see?

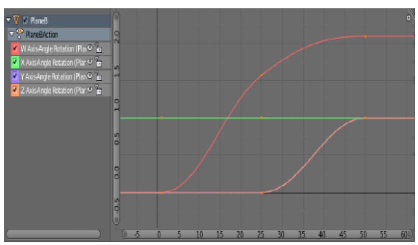


Figure 14. Axis-angle rotations: animation curves.

Yeah, right. Just curves. It's actually very difficult to know how they translate visually. While with Eulers we could grasp the meaning of the F-curves, now it's difficult to tell how the object is rotated.

Wouldn't it be nice, however, to have a rotation system which, while keeping its immunity to gimbal lock, at the same time produced perfect and smooth rotation interpolations, and not just fixed-axis ones?

Yeah, that would be awesome...!

Quaternion rotations

Quaternions were discovered by the Irish mathematician Sir William Rowan Hamilton.

According to the Wikipedia, "the breakthrough finally came on Monday 16 October 1843 in Dublin, when Hamilton was on his way to the Royal Irish Academy where he was going to preside at a council meeting. While walking along the towpath of the Royal Canal with his wife, the concept behind quaternions was taking shape in his mind. Hamilton could not resist the impulse to carve the formulae for the quaternions

$i^2 = j^2 = k^2 = ijk = -1$

into the stone of Brougham Bridge as he passed by it."

This reminds me of the day I was walking along the streets of my home town, and it came to my mind a recipe of beans with mushroom sauce. Immediately I took my chisel and hammer (I always bring them in my pockets, just in case). I couldn't resist carving the recipe into a stone of my neighbour's wall... Surprisingly, Wikipedia didn't mention that. My neighbour however, did mention it to his lawyer (he is allergic to mushrooms).

Back to quaternions, you can just forget about the formulae that Hamilton carved. Actually, you can forget about most of the maths around quaternions (unless you are a mathematician, a 3D software developer, or just very interested in Algebra).

A quaternion is a vector, i. e. a set of numbers, in a specific 4-dimensional space. In this case, this vector has four numbers. These four numbers are called X, Y, Z and W in Blender. Just check it in the Transform panel, setting Quaternion (WXYZ) rotation mode... Doesn't it remind you of something?

Of course, axis-angle has the same component names. So are these values related somehow to the corresponding axis-angle values? Absolutely. Let's see the differences, though.

In the first place, a quaternion can represent a rotation only if it is normalized, which means that the length (or modulus) of the vector must be 1 (this is called a unit vector). What does it mean in practice? Mathematically:

$$W^2 + X^2 + Y^2 + Z^2 = 1$$

This formula is not too useful for 3D artists. However it might help to understand how these four values relate to each other. In the first place, none can have an absolute value greater than 1. Second, when one value increases (in absolute value), the rest decrease, and vice versa. Absolute value means to forget about the sign, i.e., the absolute value of -0.75 is 0.75. So, all four values range from -1.0 to 1.0.

Now we know how quaternion values relate and affect each other. But what do they actually mean? Do they have the same meaning as in axis-angle? Well, actually they do. In a quaternion, X, Y and Z are still defining the same axis of rotation that axis-angle does, and W is defining an angle of rotation around that axis.

There is one unique (normalized) way to define a given rotation using a quaternion. On the other hand, in axis-angle you could define the same axis using many different combination of values, as the vector representing that axis didn't have to be normalized. Bear in mind though, that even in Axis Angle mode the rotation manipulator and the R hotkey also normalize the (X,Y,Z) vector.

Another question arises here; what units is W using to describe an angle, as it can only range from -1.0 to 1.0? To understand the correspondence between the axis-angle W value and the quaternion W value, we will call the first AW, and the second QW. Its relation is as follows:

$$QW = \cos(AW / 2)$$

If you know what a cosine function is, great. If not, don't worry the slightest bit. The only thing you should be aware of is how quaternion W behaves in relation to axis-angle W (the actual angle of rotation around the axis). The following table has a few examples that might help you:

Quaternion W	Angle in radians	Angle in degrees
1.000	0	0
0.707	$\pi / 2$	90
0.000	π	180
-0.707	$3 \times \pi / 2$	270
-1.000	$2 \times \pi$	360

You could think after seeing this table, that if a quaternion with W=1 is equivalent to a 0º angle, and with W=0 it represents 180º, then 90º should correspond to W=0.5. Actually it doesn't work like this, as you can see in the table, as the cosine doesn't behave like a linear function.

It actually behaves in a more "circular" way, which is much more suitable for rotations. Observing the table, you can think there is no way to use a quaternion to define a rotation beyond 360º, or below 0º. You think well. This is a small drawback of quaternions, but later we will see how to overcome it.

So let's try to see how a quaternion works. Open 'RotationsWorld.blend' once more and select one of the airplanes. Set rotation mode to Quaternion (WXYZ). Initially, W has value 1.0, which means a rotation of 0º, so we don't need any rotation axis. It doesn't matter then if X, Y and Z are all zero.

Now increase the value of X slightly, clicking on the right triangle in the X slider of the Transform panel. See what happens. We have just defined an axis; a point in the direction of the X axis defines the X axis itself. If we keep increasing the X value, we are still defining the same axis, however W decreases. The bigger the value of X, the smaller the value of W. In other words, we are rotating around X axis, as W keeps decreasing towards 0, i. e. towards 180º (see the table). When X reaches 1, W is 0, which means a rotation of 180º around the X axis.

So the effect of increasing the value of X is to bring the object to this position; upside down around the X axis. Now clear the rotation. You can repeat the same experiment with Y and Z values. As you will see, all of them try to bring the object upside down around their own axis.

On the other hand, what is the effect of making W bigger? Obviously to take the object away from those upside down positions, and preserve the original position with no rotations at all. The balance between the four values is what defines the final rotation.

If you repeat the experiment using negative values, you will see the same effect but in the opposite rotation direction. Take for instance the experiment around the X axis, but this time taking it slowly towards -1.0. We are defining the same rotation values (W is still positive) but applied around an axis that runs along the X axis in the opposite direction. This is similar to what happened with axis-angle. In this case also, changing the sign of all four values has no effect on the final rotation. And with quaternions, it doesn't have an effect on animation interpolations either.

Even if in theory W cannot hold a number corresponding to a negative angle, changing its sign works in a similar fashion. For instance, W=0.707 represents a 90º rotation, while W=-0.707 is 270º, which is in fact equivalent to -90º (270º=360º-90º).

Now that we know what a quaternion is and how it works, we are ready to repeat the airshow. Set the three keyframes once more using Quaternion (WXYZ) mode. What happens now?

An incredible aerobatic manoeuvre. The audience is shouting, jumping, hugging, laughing...! The best show ever! And all thanks to Sir Hamilton and his magic chisel...

What about quaternion animation curves? Take a look at them (figure 15)... What do you think? Yeah, awful. Forget about animating those evil f-curves... And there is more. In quaternion f-curves, Linear Extrapolation doesn't work well. Since the quaternion must be normalized, its values can't keep growing forever. The relation between the four values ends up reaching a normalized balance, and so the rotation slowly stops at that point.

You have seen the main advantage of quaternions: its absolute smoothness and perfection in interpolations,

no gimbal lock, no weird movements, etc. However, we have seen a drawback: the inability to define more than one revolution, or negative angles. Let's see an example. Open a Blender scene and take any unrotated object (our good old airplane will do).

Choose Quaternion (WXYZ) rotation mode. Set a keyframe in frame 1, without rotation. Go to frame 25. Rotate it 200º counterclockwise around Z axis. Set the rotation key. Check the animation. What happens? Blender actually interpolates using a clockwise rotation!

Blender has chosen the shortest path between 0º and 200º, which is equivalent to -160º. Quaternions can't define revolutions (successive spins around an axis). They just define orientations in space; from 0º to 360º, where 0º is equivalent to 360º (value 1 is equivalent to -1 for X, Y, Z and W parameters). Blender always performs the shortest path rotation between one orientation and the next if you use the rotation manipulators or hotkey R. This means you can't rotate 180º or more between two keyframes using those. But you can get bigger angles by directly editing the Transform panel values. However you will never get a 360º or bigger rotation.

If you want to overcome this, and make the object spin several times, you have to set intermediate keyframes between the initial and final state, so that turns between them are smaller than 360º (or -360º for clockwise rotation). If you use manipulators, turns must be less than 180º (or -180º).

Gimbals and locks

No, we are not going to talk about gimbal lock anymore. Just about gimbals, and component locks in either quaternion and axis-angle rotations.

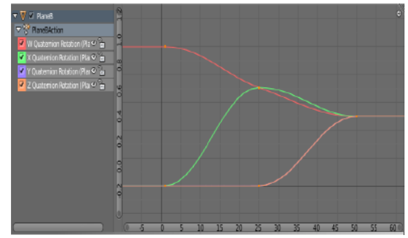


Figure 15. Quaternion rotations: animation curves.

Provided that neither of these two systems use Euler gimbals, what is the meaning of the Gimbal orientation of the 3D manipulators?

In Axis Angle, you will see that Gimbal aligns its Z component with the defined axis (X,Y,Z), so that if you rotate the manipulator blue ring (Z axis) you will be directly controlling the W value, and just the W value. However, if you want negative values, or values beyond 2π, you must edit the W value in the Transform panel.

On the other hand, when using quaternions you can see that Gimbal currently has no special meaning and is equivalent to the Local orientation. Perhaps future releases of Blender will give it a special use.

Regarding the lock buttons in the Transform panel, their use is to restrict rotations (and locations/scaling) to only the desired axes using the 3D manipulators or hotkey **R**. However, in the specific case of rotations, if you activate the **4L** button, you can restrict rotations by axis-angle or quaternion component instead of axis.

As they have 4 components (X, Y, Z, W), you get an extra lock. However, in the specific case of quaternions, remember that even changing just one of the components will affect the other three as the final vector must be normalized.

Summary

- Local or global rotation systems aren't valid for computing rotations as the order of rotation around the three axes affects the final result.
- Euler rotation systems use a hierarchy of rotation axes which is valid to compute rotations, as the three rotation components are independent. However they suffer from gimbal lock in certain circumstances.
- Axis-angle doesn't suffer from gimbal lock, but its use is almost specific to revolving around a fixed axis.
- Quaternion system doesn't suffer from gimbal lock, and interpolates perfectly any pair of orientations. However it can't define successive revolutions unless we insert intermediate keyframes in between. As it's a perfect way to define orientations in space, it is very suitable for bones animation.
- Regarding animation curves, the Euler system is the only one that provides an easy and intuitive way to edit them.

And finally...

There are a couple videos around there that might help you see rotations in action. Check the Guerrilla CG Project website (guerrillacg.org). Watch the following videos: The Rotation Problem, and Euler Rotations Explained. One warning, though: in the first of these videos there is a small mistake; whenever 'Quaternion' is mentioned, it should actually say 'Axis Angle'.

There are other 3D software packages out there that use other rotation systems, like the Heading/Pitch/Bank (or Yaw/Pitch/Roll) angles, used with the so called Tait-Bryan or cardan angles, which are a different kind of Euler rotations. But this stuff is out of the scope of this article as Blender doesn't use them ■

I hope not to have made your head rotate too much.

Be good!

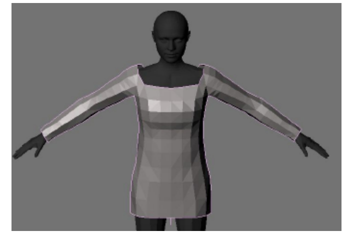
by- Sally Olle
(Estelle Parnell
Designs)

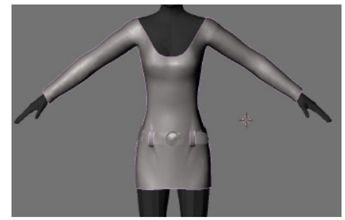
Introduction

The most advanced online 3D virtual world to hit the market, Blue Mars features photo-realistic rendering with CryENGINE-2 by CryTek and motion-captured avatar animations. Blue Mars launched in open beta to players and developers in September 2009. In about a year, the number of completely terraformed Cities, Villages and Metropolises (the basic real-estate categories on Blue Mars), has increased tenfold and a dedicated community of users and developers has been established.

As such, Blue Mars has a very attractive emerging economy. The BLU$, or Blue Dollar, is the Blue Mars currency, and it is easily redeemable via each developers' PayPal account. I was drawn to Blue Mars by the superior graphics, and most of all by the versatility and realism of the mesh clothing, having previously been a clothing designer in Second Life.

The good news is that content for Blue Mars is made in 3rd party software such as Blender, which will give existing Blender users a fantastic head start into creating content for this platform. Content is imported into Blue Mars using the Collada format. There is a freely available Blender plug-in for Collada. The Collada file is imported into the relevant Blue Mars editor (editors exist for clothing, furniture, bodies, Cities etc) where textures, maps and specialised shaders are applied to the content, and it is packed for uploading to Blue Mars.



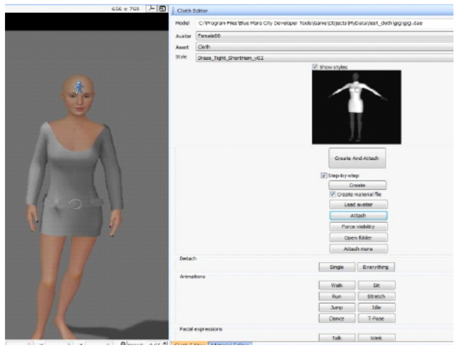


Creators can register as developers and download the developer toolkit for free at www.bluemars.com.

Creators can sell content in rented "shops", with the shop editor enabling them to customise the shop interior to their liking. Developers can also rent vacant blocks in Cities where they can create external shop structures for their own use.

These images depict the workflow for a simple retro dress from Blender to Blue Mars.

For beginners, the Blue Mars editors include a number of cloth templates to help get you started, though you can create any mesh from scratch. In this case I trimmed the mesh to the desired shape, and sculpted it to fit the Blue Mars reference avatar better.

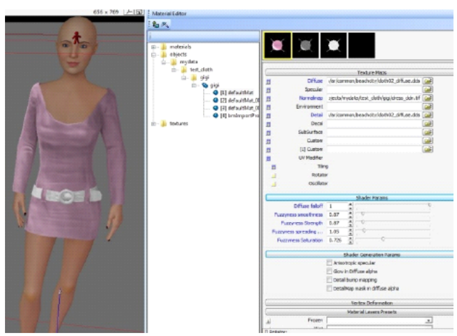


I exported the mesh from blender as a Collada file, and imported it into the Blue Mars cloth editor.

Here I applied textures and normal maps to the different linked objects, and selected the cloth shader.

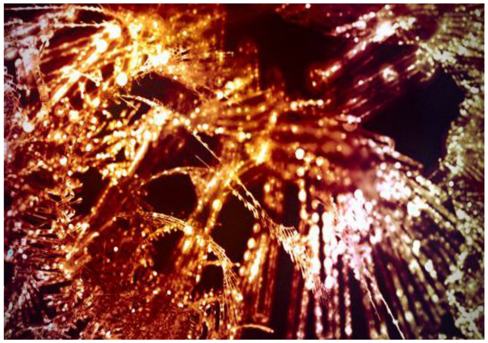
I then packed the object for uploading and went to the developer web page to upload the packed file, and to set a description and price for the object.

Once uploaded, the content enters the Blue Mars QA Process. When it is released, the designer may allocate the new item to one of their shop shelves for sale.



With a small population, the financial rewards are not yet great, but there is a tangible growth in the market and my personal enjoyment of creating cloth cannot be denied. I am sure there are great things to come for Blue Mars as it develops from its beta status and I certainly want to be there to watch it grow.

Estelle Parnall is the avatar behind the Australian content creator Sally Olle. She has been an active developer in Blue Mars since April 2010 and owns the Blue Mars city Fashion Esplanade where she sells a variety of content. www.estelleparnall.com



By - Robert J. Tiess

Introduction

Blender has a virtually limitless reservoir of untapped potential as either an artistic or technical tool. Many exciting possibilities can arise when you experiment with Blender's capabilities and try different approaches to creating and rendering scenes.

Many of you might remember my "Astrobiology" series of images posted on BlenderArtists.org. Those works were exciting and educational for me because they helped me push both my artistic and technical skills further while reaffirming to me how incredibly flexible Blender will always be as a tool for expression, imagination, and exploration.

For the images I submitted to this issue of BlenderArt Magazine, I wanted to do something different. Rather than tapping the techniques I developed for those projects, my intent was to focus more on using Blender's internal texturing system to achieve interesting details and visual possibilities.

At first glance Blender's procedural texture set might seem underwhelming and incapable of anything complex or intriguing. I tended to think that way in my earliest days learning Blender. Back then there were no texture nodes, no render nodes, just texture slots in materials.

While the main mesh of this tutorial's image (Microlifeform 4) uses four texture slots, and while it is true we can achieve more complex and interesting results by stacking textures, we are not limiting ourselves to that very useful and flexible technique. We are also going to make use of the material's Alpha (transparency) setting and pursue more of a "volumetric" texture.

How will we do this? Nested meshes: several meshes, each inside the other.

- Step one is to place a mesh in the scene. In this example I have opted for a Sphere. I reshaped, resized, and rotated the sphere in my project (defined at 1024x1024 pixels).
- Next, I added an Empty object to the scene as we are going to use this Empty to help us resize the Sphere recursively (over a series of repetitions). We need to move the Empty to where the Sphere is. To do this we copy the Location and Rotation of the Sphere by first selecting the Empty, then SHIFT key + selecting the Sphere.
- Having selected both objects, I pressed the CTRL and C keys to make the Copy Attributes menu appear, selecting Location. CTRL and C keys were pressed again to then copy the Rotation of the Sphere.
- Next, I selected the Sphere and added an Array modifier to it. We are going to use the Object offset and use the Empty as the object the modifier uses to generate successive meshes. How many? For this example I specified a Fixed Count of 20 in the Array modifier.
- After doing this, I selected the Empty object and resized it (S key) by manually entering a value of .995. If all has gone well up to this point, you should see multiple spheres within each other.

The next phase involves material and textures. With the Sphere selected we add a material. The material needs ZTransp (transparency) activated. I selected an Alpha value of .500. I didn't want any specularity, so those values are minimized.

In the texture slots are three Cloud textures. They have relatively small Noise sizes (ranging .300 to .500). A Stucci texture is also used in the second texture channel. Together three of these four textures are set to affect the material's Alpha setting.

- Two textures use a Subtract blend mode, while one blend mode is set to Add.
- Nor (bump) and Col (color) are also affected by the textures in various ways.
- The scene is lit by three lamps: one Sun lamp (ray tracing), one Hemi (hemispheric) lamp, and one omnidirection Lamp.
- Three World texture settings also contribute slightly to the ambiance of the scene.

There is something else going on in the example: particles. They serve to establish the surrounding "cilia" of this imaginary microscopic life form. For this we only need a ring of vertices surrounding the sphere.

These vertices are assigned a simple Lambert / Blinn material with a Blend texture used to fade out the tips of the emitted hair particles.

The major particle settings are: Hair (particle system type), 222 particles, Normal value of .400, Random value of .200, Brown (brownian motion) value of 8.00, Damp (dampening) value of .800, B-Spline interpolation, and Strand Render.

In the microlifeform4-example.blend file you will notice there is in fact only half a sphere. This was done to speed up render time. Textures and meshes used as they are here result in very long renders, so this is one way to achieve our desired result without forcing Blender to calculate mesh faces which would not, in this project, make any difference in the final outcome.

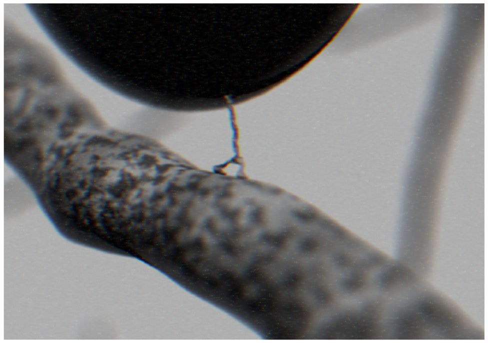
You might also notice some Render Nodes. These help us use Blender to maximize the potential of the final output image. There's some defocusing for depth of field, RGB Curves, and some other nodes used to tweak the final result all within Blender.

The example file is provided in hopes of encouraging you to experiment with different settings (material, textures, lighting, render nodes, etc.). Change values and see what happens as a result.

In fact, I think it's not only useful but necessary to allow yourself some time to use Blender in exploratory and unpredictable ways. You can learn more about Blender's and your own creative capabilities and pleasantly surprise yourself!

Technical notes

Although this project was created in Blender 2.49, it, as well as the techniques referenced in the tutorial, work in the latest Blender 2.55 beta. Render times for the project file will be lengthy even on fast computers, so patience is a must



By - Enrique Sahagún

Introduction

One of the most amazing stories I heard when I studied biology was about the way the cell distributes material within itself. The same net of microtubules that sustains the structure of the animal cell serves as a railroad to transport food and building materials. But the best is yet to come. The ones responsible for this transport are a family of small, two-legged, funny proteins called Kinesins [1]. Recent studies show that they actually walk along the microtubules while carrying the materials inside big vesicles located near the top of the kinesin molecule.

First approach

The goal was to fake a video in which a kinesin could be seen carrying a huge vesicle [2]. In principle I was concerned about the taste of the final images and not about being precise in a biological sense (shame on me!). It is possible to model the kinesin using the data from the PDB (Protein Databank) [3] using a script by Michael Gantenbrinker [4]. Instead, I decided to make a roughly similar model with a simple armature. The movements of the kinesin were made slow to simulate both the absence of gravity and the erratic flows of fluids within the cell.

The microtubule was modelled to be a kind of organic pipe and is not realistic either. The space was filled with a bunch of moving bubbles to simulate the cell environment which in reality happens to be much denser. These bubbles were animated using Blender physics. Finally, the camera was animated using a shaky effect [5].

The light is made of two sun lights with no shadows. It is important to realize that in micrographs, darker areas can be mistaken for shadows. Depending on the technique, darkness depends on the density of materials, or on the angle the faces of the objects are pointing.

There are mainly two kinds of textures in this project. The ones for the kinesin and microtubules are simple materials with no specular or mirror properties. Remember that at this scale, mirroring or specularity makes no sense. Their textures are cloudy textures with slight normals. The bubbles that simulate the environment have a transparent texture with a high IOR.

Node editor

Once the kinesin was animated, it was time to start working with the node editor. First of all I set a strong defocus filter with a variable depth of field. Then I added a noise layer which was blurred with a blur filter. To finish, something that in principle is not a typical micrograph artifact but which works very well in the final scene: a lens distortion filter with variable dispersion.

And that's all. Although, as I have said, many elements in this construction are not realistic, the final results gives a nice feeling of a living micrograph [6].

[1] http://en.wikipedia.org/wiki

[2] http://valelab.ucsf.edu

[3] http://www.pdb.org/pdb/home

[4] http://wiki.blender.org/index.php

[5] http://mke3.net/projects/bpython

[6] http://vimeo.com/12486048

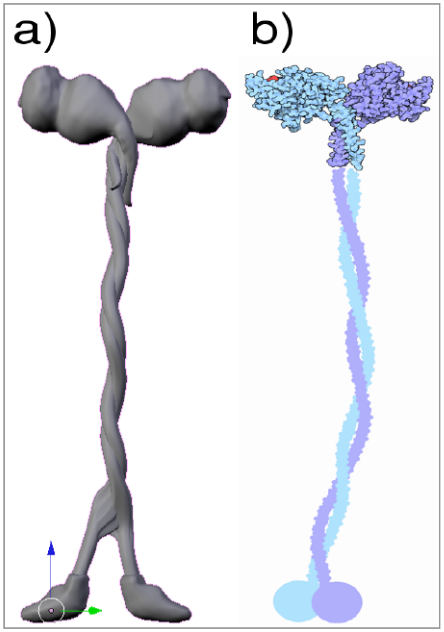
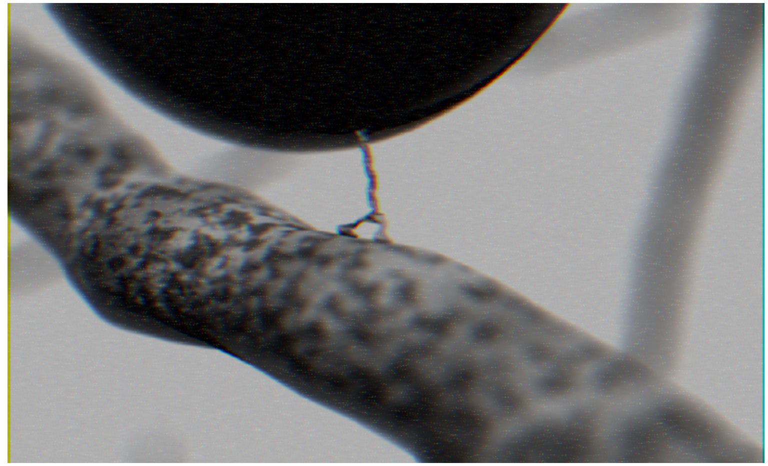
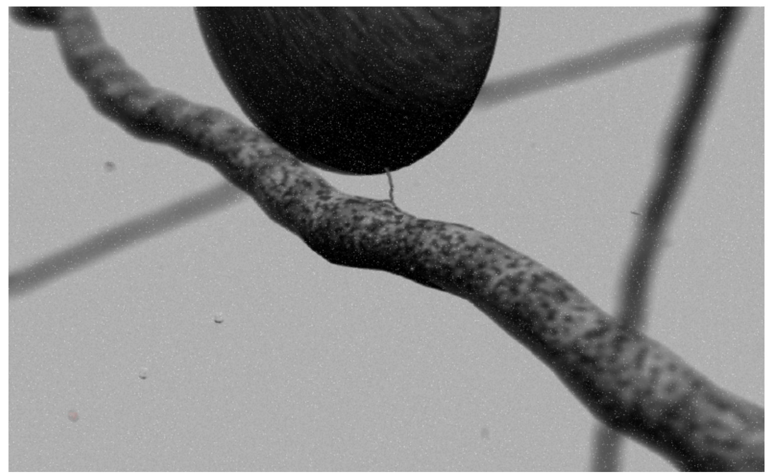


Figure a) Kinesin model. b) Kinesins (the one shown is from PDB 3kin)





By - Rod Cockcroft

Rod Cockcroft has extended the facilities at www.blendercomic.com (from a Blender comic directory) to include graphic novels created with or partially with Blender and has started a new Sintel story.

In the forums people can collaborate with others to create or discuss stories. Links can be created from each scene in a story directly to the forum so each scene can be discussed easily. If you think you can create a better storyline than one that has already been created, a fork can be inserted to create a new storyline.

There are three forums for writing and discussing stories:-

Creative commons stories

All the content in this section must be Creative Commons

Mixed copyright stories

People can contribute to stories in this section and include copyright

restrictions on their work while including content with a Creative Commons

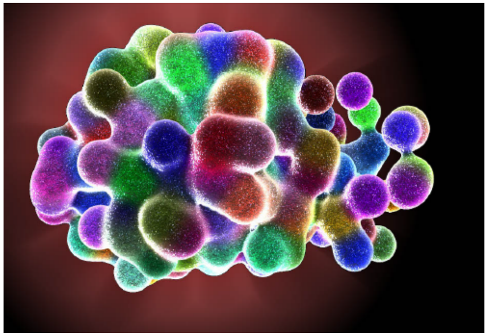
License.

Copyright protected stories

This section is for stories that are completely copyright protected.

Anyone could include a story that they have completed themselves or could develop a story in a private forum with only invited members in their group.

Rod has put the new Sintel story in the Mixed Copyright Stories section. If anyone would like to join in or start a new story visit www.blendercomic.com



By -
Raluca Mihaela Andrei,
Mike Pan
and Monica Zoppè

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

Introduction

Biologists know that, if the information of life is stored and transmitted through nucleic acids (DNA and RNA), the processes that do the actual work are most of the times proteins. These are active in all aspects of life, and in the latest years we are starting to get a glimpse of how they work. Proteins are machines composed of amino acids, which are in turn small groups of atoms arranged in specific ways[1]. Scientists are obtaining more and more information on the 3D arrangement of such atoms, and are starting to understand their activity through motion.

On the basis of information obtained by experiments of nuclear magnetic resonance (NMR), 3D visualization tools provided by BioBlender allow biologists to build a reasonable sequence of movement for proteins. It also includes a dedicated visual code to represent important features of their surface (Electric and lipophilic potential) on the protein itself, using photo realistic rendering and special effects.

BioBlender is a software extension of Blender 2.5[2], an interface for biological visualization that allows the user to import and interactively view and manipulate proteins. It was developed and is maintained by the Scientific Visualization Unit of the CNR of Italy in Pisa, with the help and contribution of several members of the Blender community. Material, scenes, publications and other relevant information can be found at www.BioBlender.net and/or www.scivis.ifc.cnr.it.

BioBlender for Windows is available from www.bioblender.net (on Linux machines it can be used with Wine). Because of its specialized nature, it requires the installation of PyMOL[3.4] , Python 2.6 [5] and NumPy[6] , which are all provided in Installer folder from the downloaded package.

Using BioBlender to build an animation

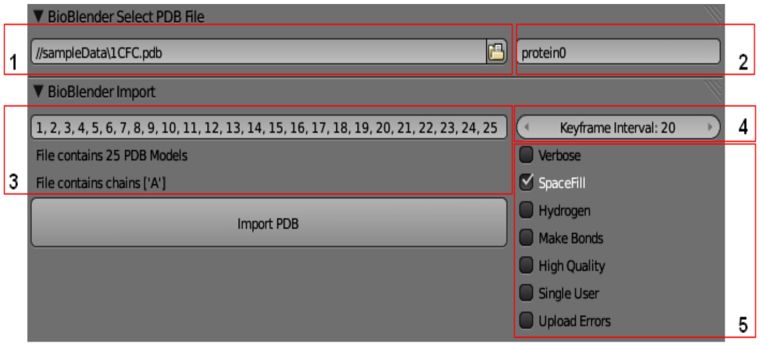
To start BioBlender, simply go to the Bin folder and launch blender.exe, then open the **template.blend** scene (stored in BioBlender folder).

Notice that the template file not only has an optimized user-interface layout for biologists, but the template scene also contains lights, camera and world settings that are ideal for visualizing molecules. This setup ensures that researchers who are not familiar with the 3D software can still effectively use BioBlender. Each interface element (buttons, sliders, toggles) has help text associated with it. By placing the mouse over them a pop-up text describes the function. Errors and progresses are displayed in the console. Critical errors will appear in the main BioBlender as a pop-up under the mouse. The atoms size is of order of Ångström (Å), therefore the scale used is 1 Blender Unit = 1 Å.

This tutorial assumes that you already have BioBlender downloaded on your computer, with the required programs installed.

1. Select and import a .pdb file

PDB files contain a description of one or multiple conformations (positions) of a single molecule. Different conformations of the same protein are listed in one NMR file and are called MODEL 1, MODEL 2 etc.



In the **BioBlender Select PDB** File panel:

- Select the .pdb file by browsing from your data (**1** in figure). The file included in sampleData folder contains the 25 models of Calmodulin [7]. Alternatively, simply type the 4-letter code for the .pdb file to be fetched from www.pdb.org [8] (make sure to pick an NMR file);
- Change the name of the protein (by default it is named "protein0") in the field on the right (**2** in figure). Naming the proteins is just a good habit that will help keeping the scene organized. Once a file is selected, the number of models and the chains are detected and shown in the BioBlender Import field (**3** in figure);
- Choose 2 models to import in the scene (by default all models are listed) typing their number separated by comma;
- In the **Keyframe Interval** slider (**4** in figure) set the number of frames between the protein conformations (Min 1, Max 200).

A list of options are available to be considered before importing the protein in the Blender scene (**5** in figure):

Verbose: enable to display in the console extra information for debugging;

SpaceFill: enable or disable to display the atoms with Van der Waals or covalent radii in the 3D scene, respectively;

Hydrogen: enable to import Hydrogens if they are present in the .pdb file. This option makes importing much slower and it is important only for visualization. If the .pdb file does not contain Hydrogens (or if you chose not to import them), they will be added during the Electrostatic Potential calculation using external software;

Make Bonds: enable it to have atoms connected by chemical bonds. Despite being time consuming this operation is very important in motion calculation;

High quality: displays high-quality atom and surface geometries; slow when enabled;

Single User: enable to use shared mesh for atoms in Game Engine; slow when enabled;

Upload Errors: enable to send us automatically and anonymously an email with the errors you generate. This makes us aware of the problems that arise and help us fix them.

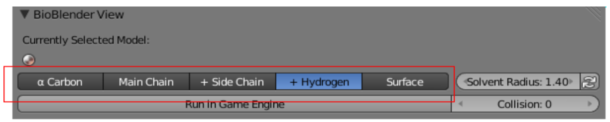
Finally, press **Import PDB** button to import the protein to the 3D scene of Blender. Blender displays the protein in motion (by linear interpolation between atoms in the conformations; Esc to stop the animation).

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

2. Visualization in the 3D viewport

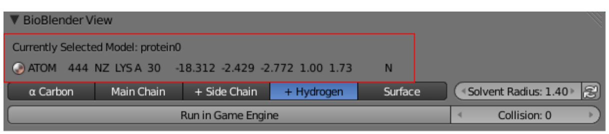
Once imported, the protein is displayed with all atoms, Hydrogens included (if the Hydrogens check-box was enabled). The first 4 buttons in the **BioBlender View** enable different views: only alpha Carbons, main chain (N, CA, C), main chain and side chains (no H), or all atoms.



If the **Surface** display mode is selected, BioBlender will compute the surface of the protein by invoking PyMOL software, an external application. It uses the **Solvent Radius** set by the user and returns the Connolly mesh [9], displayed on the BioBlender 3D view. The default radius (1.4 Å) is the standard probe sphere, equivalent to water molecules.

To check the appearance of surface calculated with different solvent radii, change the solvent radius value and press refresh button. The current surface is deleted and a new one is created.

When atoms are displayed, by selecting one atom in the 3D display, the protein information of the selected atom is printed in the area outlined below; in the 3D view the selection will extend to the other atoms of he corresponding aminoacid.



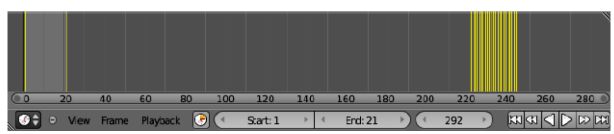
3. Protein motion using the physic engine

To calculate the transition of the protein between the 2 conformations the Blender Physics Engine is used. Press **Run in Game Engine** button to see the transition. Press Esc to leave GE and then 0 on Numerical Board to see from the camera point of view.

Hit **Run in Game Engine** button again for an interactively view. When inside the Game Engine, the mouse controls the rotation of the protein, allowing to inspect the protein from all angles. The also applies an ambient occlusion filter to the scene, giving the viewer a much better sense of depth.

Set the **Collision** mode to one of the following states: 0, 1 or 2. When set to 0 the transition between the conformations is done using linear interpolation; the atoms will simply move from one position to the other. When set to 1 the collisions between atoms are considered, resulting in a more physico-chemical accurate simulation[10].

When set to 2, the newly evaluated movement will be record to F-Curves. Go to the Timeline panel on Blender and see that the new conformations are recorded at different time (200 frames away from the last model imported) as shown in the figure below; in this way both sets of transitions are available for comparison. These conformations can be exported as described later in section 6.



Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

4. Molecular Lipophilic Potential Visualization

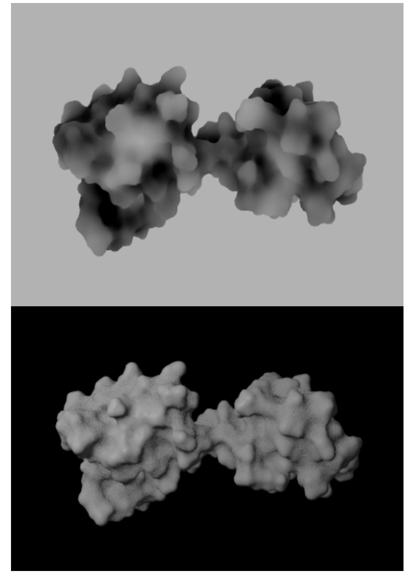
This visualization method is a novel way to see the MLP values of a protein onto the surface. Normally this is a relatively time consuming and tedious process involving running different programs from the command line, but BioBlender simplifies the entire process by allowing the user to do everything under one unified interface.

In **BioBlender MLP Visualization** section:

- Choose a **Formula** (**1** in figure; Testa formula [11] is set by default);
- Set the Grid Spacing (2 in figure; expressed in Å, lower is more accurate but slower) for MLP calculation;
- Press **Show MLP on Surface**. It may take some time as the MLP is calculated in every point of the grid in the protein space, then mapped on the surface of the protein and finally visualized as levels of grey (light areas for hydrophobic and dark areas for hydrophilic [12]).

A typical protein has varying degrees of lipophilicity distributed on its surface, as shown here for CaM.

Use **Contrast** and **Brightness** sliders to enhance the MLP representation of your protein. Once you are satisfied with the grey-levels visualization hit **Render MLP to Surface** button for the photorealistic render. This process is also time consuming and it always refers to last changes in the MLP grey-levels visualization. When the calculation is done (the button is released) press F12 on your keyboard.



Note:This is the MLP representation using our novel code: a range of visual features that goes from shiny-smooth surfaces for hydrophobic areas to dull-rough surfaces for hydrophilic ones. The levels of grey are baked as image texture that is mapped on specular of the material. A second image is created by adding noise to the first one and map it on bump. The light areas become shiny and smooth while the dark ones dull and rough as shown in the figure.

Press Esc to go back to the Blender scene.

5. Electrostatic Potential Visualization

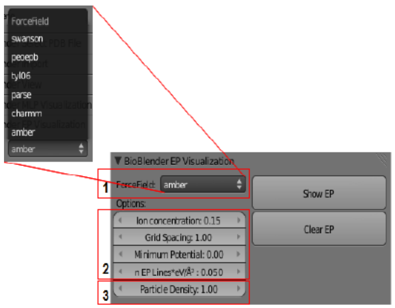
EP is represented as a series of particles flowing along field lines calculated according to the potential field due to the charges on the protein surface. For this reason, it is necessary to perform a series of steps (as described in [12]), and to decide the physical parameters to be used in the calculation (2 in the figure).

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

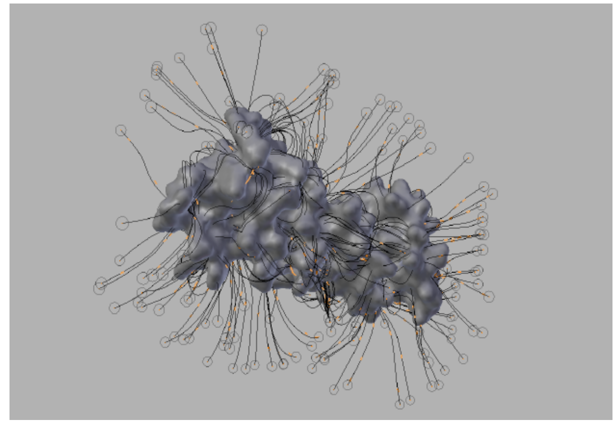
In **BioBlender EP Visualization section**:

- Choose a **ForceField** (**1** in figure; amber force field is set by default);
- Set the parameters for EP computation, using the options shown in the figure below:
- **Ion concentration** – 0.15 Molar is the default, physiological value;
- **Grid Spacing** – in Å, lower is more accurate but slower;
- **Minimum Potential** – the minimum value for which the field lines are calculated – the default value is 0 which implies calculation of all possible lines; increase it if you want to enhance the representation of EP;
- **n EP lines*eV/Å2** – the number of field lines calculated for eV/Å2.



Now press **Show EP** button. *The process is time consuming as **Show EP** button invokes a custom software that calculates the field lines and exports them in the BioBlender 3D scene as NURBS curves. The positive end of each curve becomes an emitter. The particles flow along the curves from positive to negative.*

Change the **Particle Density** (3 in figure) to modify the number of the particles visualized in the scene. **Clear EP** to delete the curves and the emitters.



6. Output

To see the protein movement with the surface properties you have to render a movie. Since the movement implies a change of the atomic coordinates, the surface properties must be recalculated frame by frame.

In the **BioBlender Output** panel set the output file path (by default it is set to tmp folder); choose the kind of representation you prefer to render from the **Visualize** curtain menu:

- **Atom** – render only atoms;
- **Plain Surface** – render only surface;
- **MLP** – render surface with MLP;
- **EP + Plain Surface** – render surface (no MLP) and EP;
- **EP + MLP** – render surface with MLP and EP;

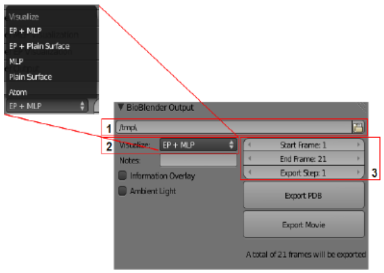
set **Start Frame** – the first frame of the animation;

Raluca Mihaela Andrei[1,2], Mike Pan[1,*] and Monica Zoppè[1§]

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

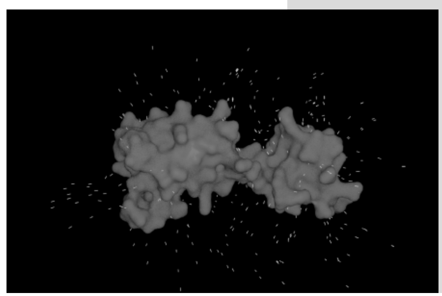
set **End Frame** – the last frame of the animation;

set **Export Step** – the number of frames to skip during export, mostly used for faster export of .pdb files; enable **Information Overlay** to print extra information on the final image; enable Ambient Light only for GE visualization; do not enable it for MLP representation as its effect is confusing for MLP visual code.



Hit **Export Movie** to render every frame of the animation. The output is a sequence of still images, this ensures that the rendering is resumed if the rendering process is disrupted. During section 3 Blender GE calculated and recorded intermediate conformations as keyframes. To save these coordinates as .pdb files for further analysis using external software, press **Export PDB**. A .pdb file is saved for each frame in the selected output.

To obtain the movie follow standard Blender procedures: open the Video Sequencer Editor: **Add -> Image**, select the sequence of images, go to **Properties** window and set the **Output** path and the **File Format** to **AVI JPEG** in the **Output** panel and **Start** and **End** frame in the **Dimensions** panel. Now press **Animation** button in the **Render** panel.



Now you have your protein moving with the surface properties visualized. An image of CaM with EP and MLP is shown in the image below ■

References

1 *Zoppè, M; Porozov, Y; Andrei, R M; Cianchetta, S; Zini, M F; Loni, T; Caudai, C; Callieri, M (2008) Using Blender for molecular animation and scientific representation. Proceedings of the Blender Conference Blender*

2 *DeLano, WL, The PyMOL Molecular Graphics System, 2002*

3 *The PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC*

4 *Python*

5 *NumPy*

6 *Kuboniwa H, Tjandra N, Grzesiek S, Ren H, Klee C B, Bax A (1995) Solution structure of calcium-free calmodulin. Nat Struct Biol 2: 768-76*

7 *Berman, H M; Westbrook, J; Feng, Z; Gilliland, G; Bhat, T N; Weissig, H; Shindyalov, I N; Bourne, P E (2000) The Protein Data Bank. Nucleic Acids Res 28: 235-42*

8 *Connolly, M L (1983) Solvent-accessible surfaces of proteins and nucleic acids. Science 221: 709-13*

9 *Zini, M F; Porozov, Y; Andrei, R M; Loni, T; Caudai, C; Zoppè, M (2010) Fast and Efficient All Atom Morphing of Proteins Using a Game Engine. (under review)*

10 *Testa, B; Carrupt, P A; Gaillard, P; Billois, F; Weber, P (1996) Lipophilicity in molecular modeling. Pharm Res 13: 335-43*

11 *Andrei R M, Callieri M, Zini M F, Loni T, Maraziti G, Pan M C, Zoppè, M (2010) A New Visual Code for Intuitive Representation of Surface Properties of Biomolecules. (under review) Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1∫*

12 *Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca,*

13 *Pisa, Italy*

14 *2Scuola Normale Superiore, Pisa, Italy*

15 **Present address: University of British Columbia, Vancouver, Canada Corresponding author*

Raluca Mihaela Andrei1,2, Mike Pan1,* and Monica Zoppè1§

1 Scientific Visualization Unit, Institute of Clinical Physiology, CNR of Italy, Area della Ricerca, Pisa, Italy
2 Scuola Normale Superiore, Pisa, Italy
* Present address: University of British Columbia, Vancouver, Canada
§ Corresponding author

by- Francisco M. Gomez-Campos

Introduction

Three-dimensional (3D) technologies have always seemed to relate to futuristic applications. However, 3D animation has reached maturity in the last decade, 3D movies are in fashion in cinemas, 3D television is coming next ... Let's acknowledge it: this is the present. So, why not try 3D teaching? Today, educators in schools and universities can complement their teaching with 3D animation, going beyond chalk and blackboards, overhead projectors and Power Point presentations.

This report is a summary of our developments in the Department of Electronics at Universidad de Granada (University of Granada) in the south of Spain. In the last few years we have produced educational material using Blender to do animations, to show our students some concepts in electronic physics and to help them use laboratory instruments.

We briefly describe here some technical issues regarding our videos, such as the procedure we followed to represent three-dimensional wave functions, a very important issue in Quantum Physics, and how we approached the modelling and depiction of the screen of an instrument widely used in the Electronics lab: the oscilloscope.

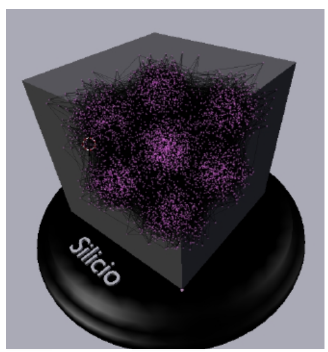


Figure 1: The mesh of a wave function in a silicon crystal.

Teaching science is a hard task today. Nevertheless, Blender gave us the chance improve communication with our students, to let them know how things work in the depths of a silicon crystal... or in the lab next to their classroom.

Blender in Quantum Physics:

It is said that Richard Feynman (1918-1988), one of the most important American physicists of the 20th century, summarized the complexity of the quantum world in a sentence: "I think I can safely say that nobody understands quantum mechanics". Nevertheless, sometimes university lecturers have to teach... quantum mechanics. From Feynman's sentence we can figure out the difficulties for students in learning something "impossible to understand"... Well, the point is that the mathematical structure describing very small things such as molecules or atoms is very complex, especially because particles are not imagined as small dots, but as something called "wave functions". These wave functions give us the probability of finding the particle at a certain point in the space. How could we draw those probabilities on a blackboard, at each point in the space? Anything that offers us support in clarifying this concept is really useful. Blender helped us with this task.

First of all, we made a program to compute these probabilities for electrons in a piece of silicon and to record it in files. They were just clouds of points. After that, we wrote a very simple script in Python to create a mesh in Blender with the dots of the files. In this way we created something like in Figure1.

In this representation we are "drawing" probabilities. Thus, in those places where there are more dots, it is more likely the electron will be found. You can see there are lobes in the wave function represented (there are a huge number of different wave functions in a piece of silicon), so there are some regions where the electron is more likely to be observed. By making a rotational movement, we can take a complete image of this piece of semiconductor and see the three-dimensional distribution of probabilities, and this is where 3D teaching starts! At least this is more enjoyable than a blackboard!

Figure 2: Drawing of the scene. For the wave function object we used a halo-type material.

Blender in Electronics:

When you buy a TV or a DVD player you always have a user guide. You are supposed to learn how it works using this guide, but you also have the device in front of you… Interaction with the instrument is crucial to learn how it works but, what happens if you just have the user guide and you can't imagine the device? You're in trouble.

The oscilloscope is a very useful instrument in electrical and electronic engineering labs. It consists of a screen where you can monitor electrical signals in a circuit. Students normally have to learn how an oscilloscope works before seeing it. This is complicated and the practice sessions take time. With Blender, you can model an oscilloscope and show in a simple way how it works. The advantage of 3D animation is that you are able to control everything in the scene, focusing the attention of the students during the explanation on those details the teacher thinks are the most relevant. And, of course, this makes science seem more fun.

We modeled a virtual lab with an oscilloscope. We tried to model the environment in a realistic way to give the impression of a serious workplace. Textures and lights resembled those found in most real laboratories in universities.


Figure 3: The virtual laboratory.

We alternated the 3D environment of the lab with the 2D scene on the screen. To carry out the modeling of the latter, we set the camera for an orthographic view. In world properties, we set a blank screen (with no signal on it) as the background texture (see Figure 4) and we added a mesh, this being the signal on the screen.

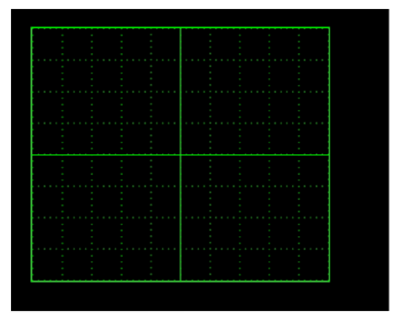


Figure 4: Blank screen of the oscilloscope

However, we wanted to let the signal appear gradually, so we used a plane with Z-transparency and moved it from one side of the screen to the other. Thus, those parts behind the plane appeared as the background image, and only part of the signal was visible. For the signal we used a halo texture. Its appearance was very close to the image on a real oscilloscope.

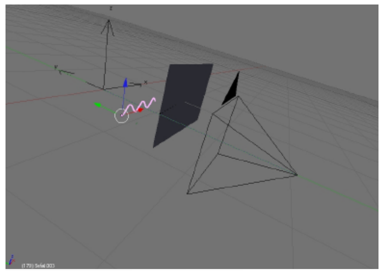


Figure 5: Diagram of the arrangement for simulating a 2D oscilloscope screen

We thought this might be useful for students in other universities, so we decided to broadcast these videos on YouTube. The number of views and the user comments are encouraging; we found there was great interest, especially from Spanish-speaking countries (the videos are in Spanish). To watch our videos, just look for user fmgomezcampos on YouTube. The core of the working team is composed of several professors with wide experience in both research and university teaching: J. E. Carceller, J. A. Jiménez-Tejada, J. A. López-Villanueva, S. Rodríguez-Bolívar, A. Godoy and myself.

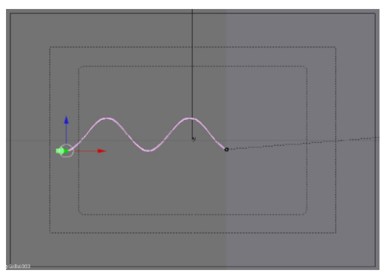


Figure 6: View of the arrangement from the camera

We thought they might also be of interest to other scientists so we submitted our works to some learning conferences, where they were a great success. And now we think it's time to let the Blender community know what we're doing in 3D teaching!

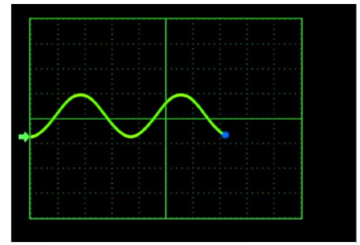


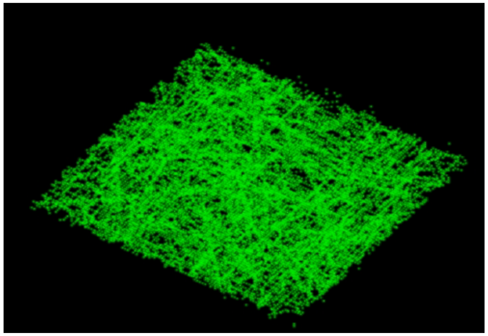
Figure 7: Rendering of the oscilloscope screen. The blue dot acts as a tracer of the signal, and is a child of the plane. The dot is placed on the edge of the plane.

Acknowledgments:

Special thanks to Monica Zoppè and her great team at the Scientific Visualization Unit in Pisa, Italy (Raluca, Stefano, Ilaria, Maria Antonietta, Claudia, Tiziana and others I did not meet but who also worked on the same project). I enjoyed meeting these great professionals and wonderful people

Francisco M. Gómez-Campos, Ph.D. in Physics and lecturer in Universidad de Granada, Spain, in the Department of Electronics. He has managed several innovative learning projects in the Universidad de Granada.

Email: fmgomez@ugr.es



By - Richard Charvat

Richard CHARVÁT 1, Eva KO? TÁKOVÁ and David LUKÁ? 2

1 Technical University of Liberec, Faculty of Art and Architecture, Atelier of Virtual Reality, Czech Republic
2 Technical University of Liberec, Faculty of Textile Engineering, Department of Nonwovens, Czech Republic

Introduction

A flow involving more than a single phase is classified as multiphase or non-homogeneous, such as liquid flows in porous fiber media. We are interested in the dynamics of the evolving interface between the distinct phases during such non-homogeneous flows in a fiber mass. The dynamics of such flow are dominated by surface tensions, porous media anisotropy and non-homogenity, fiber volume fraction, and fiber wetting behaviors.

The uncertain structural conditions in fibrous media, including the susceptibility to even small loads, as well as the tortuous connectivity of their open pores and poorly defined boundaries, result in complex local non-homogeneous flows and interfacial evolution. This complexity, in many cases, becomes prohibitive for the development of analytical theories describing these phenomena. The wetting and wicking of fiber mass constitute a class of flows that have critical scientific and first of all practical significance.

Idea

Adapt Monte-Carlo simulation based on the Ising model for a description of the wetting and wicking phenomena in fibrous media. We introduce here a 3-D Ising model, incorporated with the stochastic dynamics and the method of importance sampling, which enables us to interpret the model outputs in terms of wicking dynamics.

The essential principle of this model is based on the discretion of the whole system of a fibrous mass, a liquid source, and a wetting configuration at any given moment. The continuous media in the system, including the solid, liquid, and gas, are all divided as assemblies of individual cells occupied by the respective medium so that such a discrete system of cells can be manipulated more easily in a computer. The liquid wicking simulations are then set up from the initial configuration of the liquid layer into which the fiber mass with a predefined fiber orientation is in part vertically dipped, absorbing the liquid.

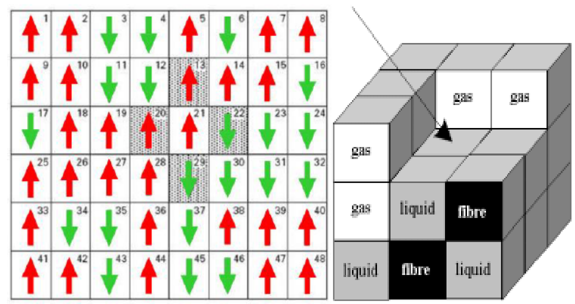


Figure 1. 2-D Ising basic ferromagnetic model vs 3-D Ising model for liquid-fiber mass interaction. A cell in the center forms a supercube with its neighboring cells. On the front surface, we can see various kinds of media that occupy the cells. For example, the white color denotes the air, the grey color denotes the liquid, and fiber cells are black.

Statistical physics in general deals with systems with many degrees of freedom. These degrees of freedom, in our case, are represented by the so called Ising variables. We assume that we know the Hamiltonian (the total internal energy) of the system.

The problem is computing the average or equilibrium macroscopic parameters observable (energy and liquid mass uptake) for a given initial system configuration. Moreover, we will monitor the kinetics or even dynamics of the system so as to simulate the wicking behavior with time, for more detail see [3],[4].

Model

The auto-model (particularly so called Ising model) and Monte-Carlo method were used especially for simulation of a liquid droplet in contact with fibrous material. The mechanism of this kind of simulation is fully described in [2].

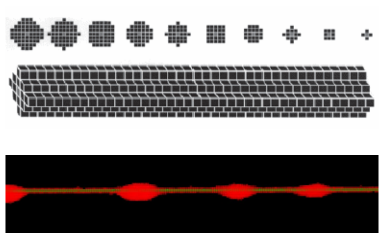


Figure 2. A simulation box with cells illustrate schematically a 3-D Ising model of droplets on single fibre in various configurations. In this case it was used for procedural modeling of the Rayleigh instability phenomena (top). Computer visualization of Rayleigh instability of liquid droplets on single fibre (bottom).

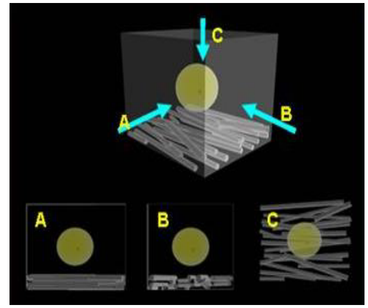


Figure 3. An illustration of the initial state of a simulated system. A dropping ball is placed on top of non-woven textile with a particular fibre orientation. After the start of the simulation process, the ball infiltrates inside, then when we change angle of wetting = dropping ball does penetrate (0°) or dropping ball doesn't penetrate (90°).

Methodology

With the use of an optimized algorithm, the 3-D Ising model improves accuracy and efficiency in simulation. This approach is capable of realistically simulating the complicated mechanisms involved in the filtration and separation processes. The fibrous material is represented by non-woven textile material.

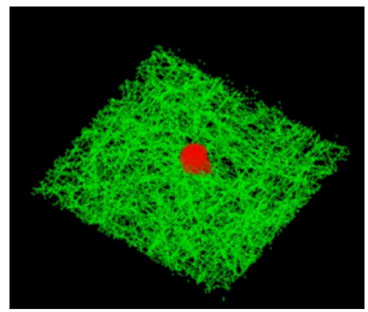


Figure 4. The result images present the equilibrium state of liquid droplet versus fibrous non-woven structure at the end of the simulation process in two ultimate states of the system: liquid with high contact angle (left) and or liquid with low contact angle (right) in contact with randomly oriented fibrous material.

Interface

Occupied elements of 3D model are imported in layers as vertices after which a volume effect is applied on them (various color depending on fibre or liquid fluid element). It is further possible to render or animate structures made by fibres with liquid interaction or even cut samples in desired positions with orthographic camera point of view. This method works properly even for very large data sets.

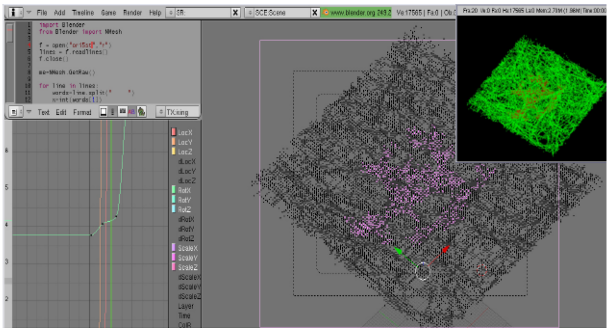


Figure 5. System of fibrous non-woven material in computer workspace. Rendered image or linear sequence of images (top right).

Eventually it will be possible to use textured slices in a voxel visualization manner, like a 3D virtual reconstruction of human body from cuts obtained by medicine computer tomography devices.

Visualization

Furthermore it is also possible to present linear or even real time content in low cost anaglyph stereoscopy or

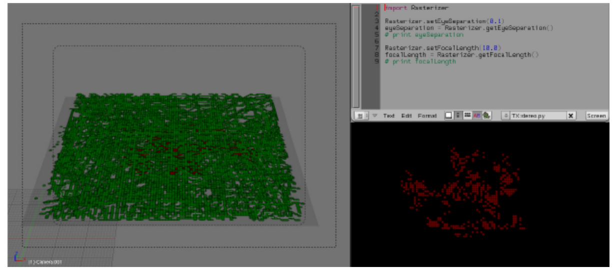


Figure 6. Reconstructed voxel slices (left) of fibrous non-woven material. Textured image slice of droplet with alpha channel (bottom right).

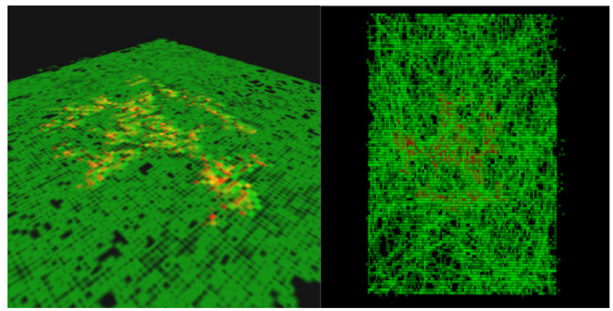


Figure 7. Perspective anaglyph of fibrous non-woven material (left). Top view of complete fibrous system with an alpha channel slice (right).

active virtual reality projection due to much better immersion.

Due to GLSL offering support for real time shaders, it is possible to experiment with scientific computing and visualization using the real time interactive game engine.

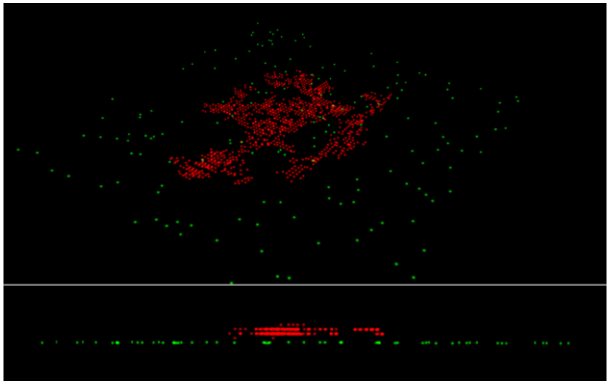


Figure 8. Real time interactive visualization via GPU. Perspective camera view (top) and side view of fibrous system with droplets (bottom).

Parallel computing architecture is a programming approach for the performing of scientific calculations on the GPU as a data parallel computing device. The programming interface allows us to implement algorithms using extensions to the standard Python language used inside Blender [1].

Authors also thank the companies Elmarco and Cummins Filtration for their support and interest in this work ■

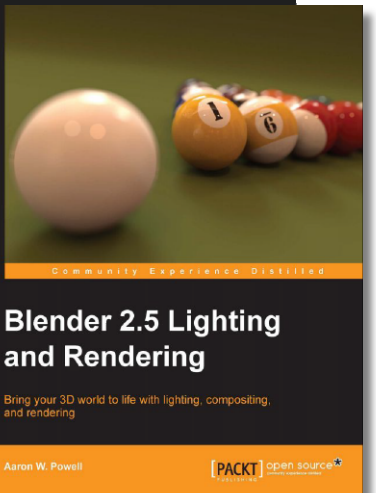
```
import GameLogic
cont = GameLogic.getCurrentController()
obj = cont.getOwner()

FragmentShader = """
        uniform sampler2D color;
        varying vec3 light_vec;
        varying vec3 normal_vec;
        void main() {
                vec3 l = normalize(light_vec);
                vec3 n = normalize(normal_vec);
                float ndotl = dot(n,l);
                gl_FragColor = texture2D(color,
gl_TexCoord[0].st)*ndotl;
        }
        """

mesh_index = 0
mesh = obj.getMesh(mesh_index)
shader = mat.getShader()
shader.setSource( FragmentShader,1)
shader.setSampler('colorMap',0)
```

Code 1. A piece of code an in integrated interpreter to apply real time pixel shader visualization via GPU.

References
1. R. Charvat: Blender Like a Nanoscope (procedural modeling), paper for 8th Annual Blender Conference in Amsterdam, 25th October 2009.
2. D. Lukas, N. Pan, A. Sarkar, M. Weng, J. Chaloupek, E. Kostakova, L. Ocheretna, P. Mikes, M. Pociute and E. Amler: Auto-Model Based Computer Simulation of Plateau-Rayleigh Instability, Physica A: Statistical Mechanics and its Applications, Volume 389, Issue11, 1 June2010, Pages 2164-2176.

3. D. Lukas, V. Soukupova, N. Pan and D. V. Parikh: Computer Simulation of 3-D Liquid Transport in Fibrous Materials, Simulation, vol. 80, issue 11, pp. 547-557, DOI: 10.1177/0037549704047307.

4. D. Lukas, E. Kostakova and A. Sakar: Computer Simulation of Moisture Transport in Fibrous Materials, Thermal and Moisture Transport in Fibrous Materials, edited by N. Pan and P. Gibson, Woodhead Publishing Limited, Cambridge, pp. 469-541, ISBN-13: 978-1-84569-057-1.



Its raining Blender books lately and Packet publishing have been on the front of this literary assault. Fortunately for us blender users it's nice time to fill up our book shelves since final re-lease of Blender 2.5 is upon us soon.

Blender is a relatively new tool for most professionals now that we have an excellent interface to begin with. We still need lots of information about crucial features of blender3d such as lighting and the material system.

Blender 2.5 Lighting and Rendering book fills that slot nicely. Although it cannot be called a complete beginners book, since you will mostly be lost on various steps if you do not have at least working knowledge of Blenders interface, it can't be called an one stop extensive book for Advance users.

So for an beginner to intermediate user, this book is easy to pickup and allows you to quickly grow on towards much more advance usage, thus making it an excellent learning companion.

The books starts off with the very basics on lighting terminologies such as color and a basic premise of color theory then gradually moving on to understanding lighting in real world settings.

After the grounds up on real life knowledge, the reader is exposed to blender’s myriad controls and features available for various types of lighting solutions. From Ambient Occlusion to Indirect Lighting, then on to outdoor and Indoor-lighting solutions. The explanation seems justly nice for newbies, but the level of explanation at some points leaves more advance users wanting for more, so clearly this book is not for experienced users.

The book offers a nice portion to Materials in Blender. even better, UV Mapping which in my opinion is a good move as it brings the user right up there with the main tool-set's of Blenders Rendering pipeline. This is supplemented with a pretty good from the grounds up of the material system and its various features namely, diffuse, spec mirrors, IOR etc.

Whats Good

- Explanations of most features, it covers almost every part of lighting and rendering including the Material system.
- Very practical with excellent exercises for practical understanding.
- Pretty straight forward and concise.
- Easy to Pick up and read.

Whats bad.

Not much really.

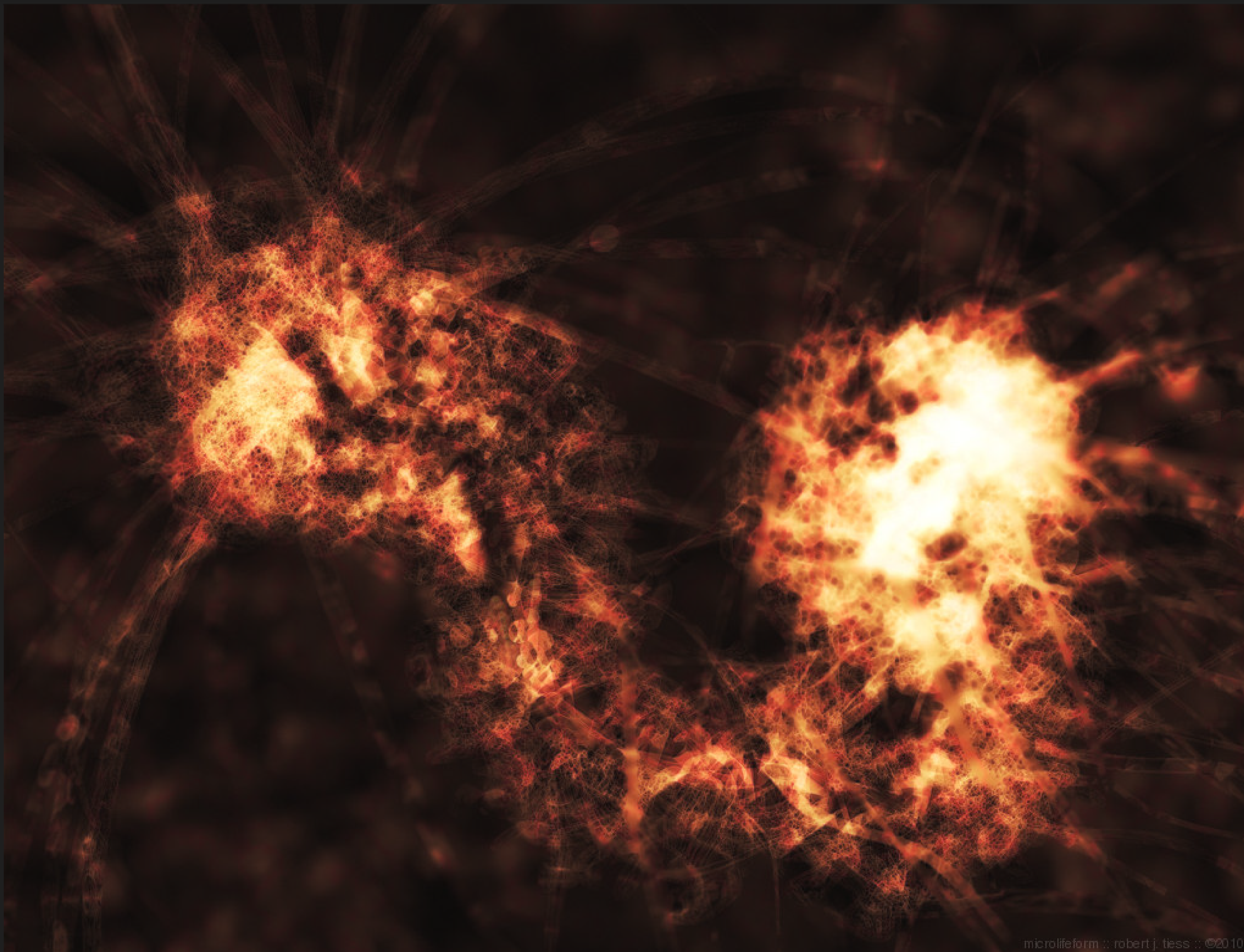
Seems like BA recommended buy ;)

Blender 2.5 Lighting and Rendering

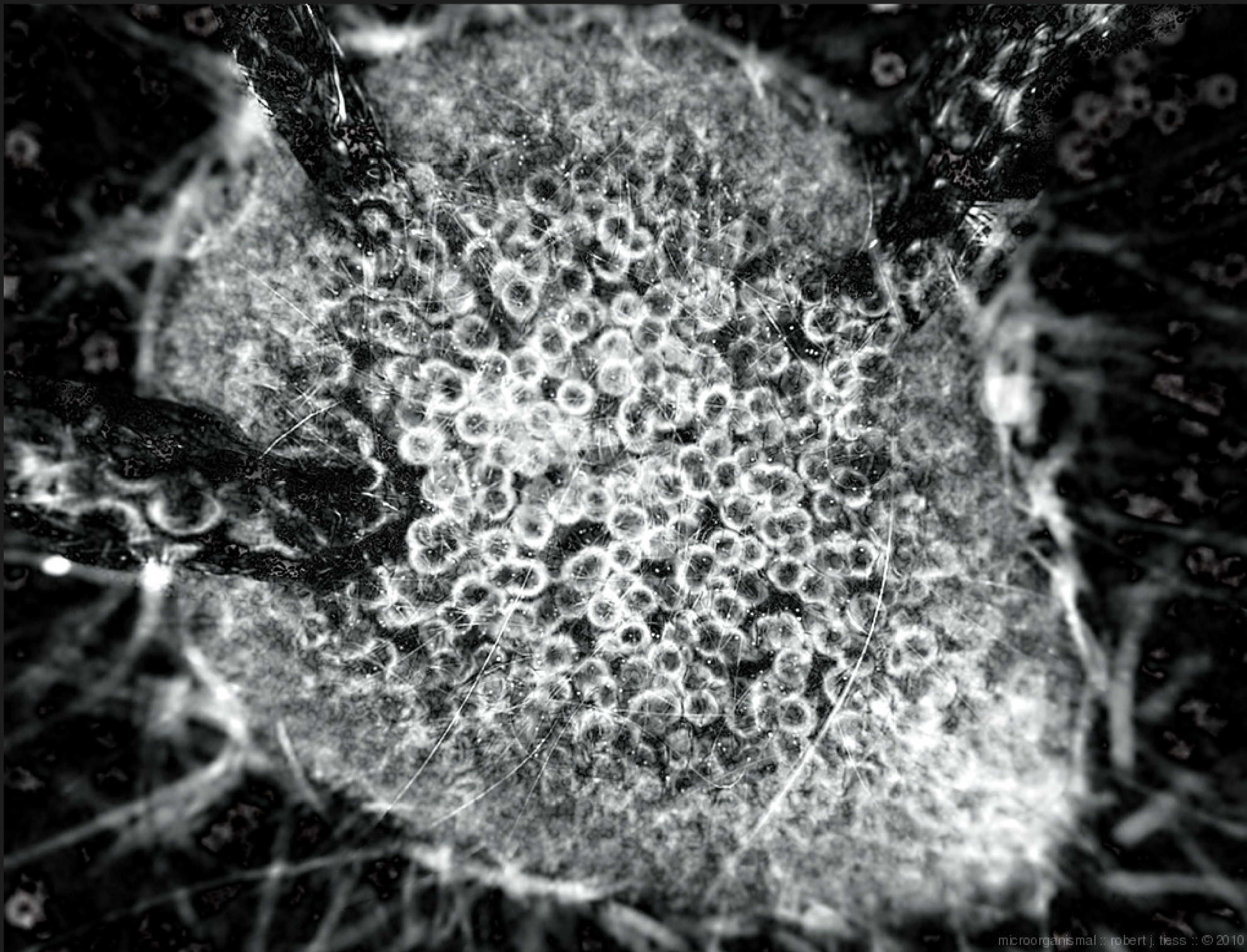
Packt Publishing
Pages 252 Approx.

ISBN 978-1-847199-88-1
www.packtpub.com

Micro lifeform - by Robert J Tiess

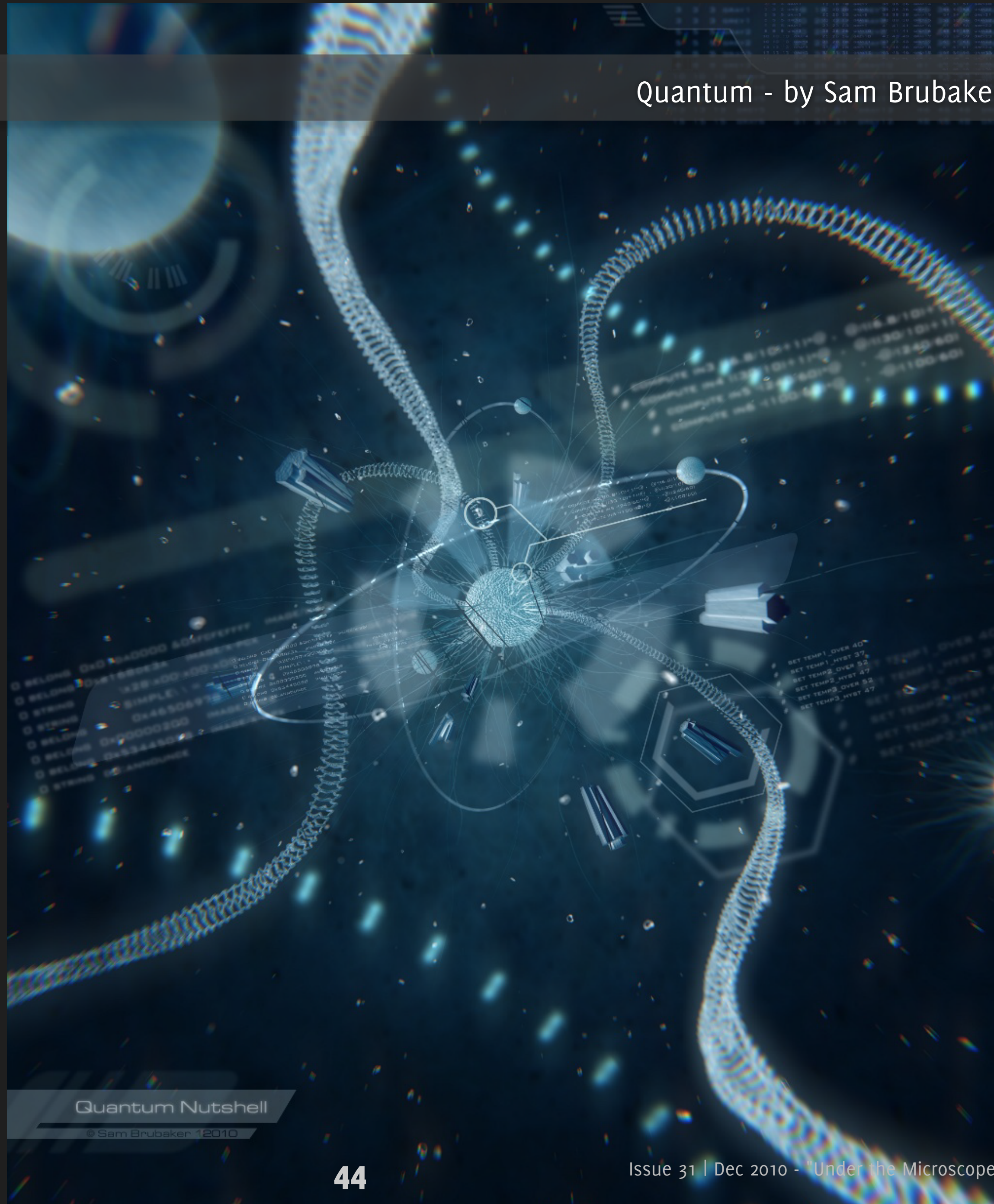


Micro lifeform - by Robert J Tiess

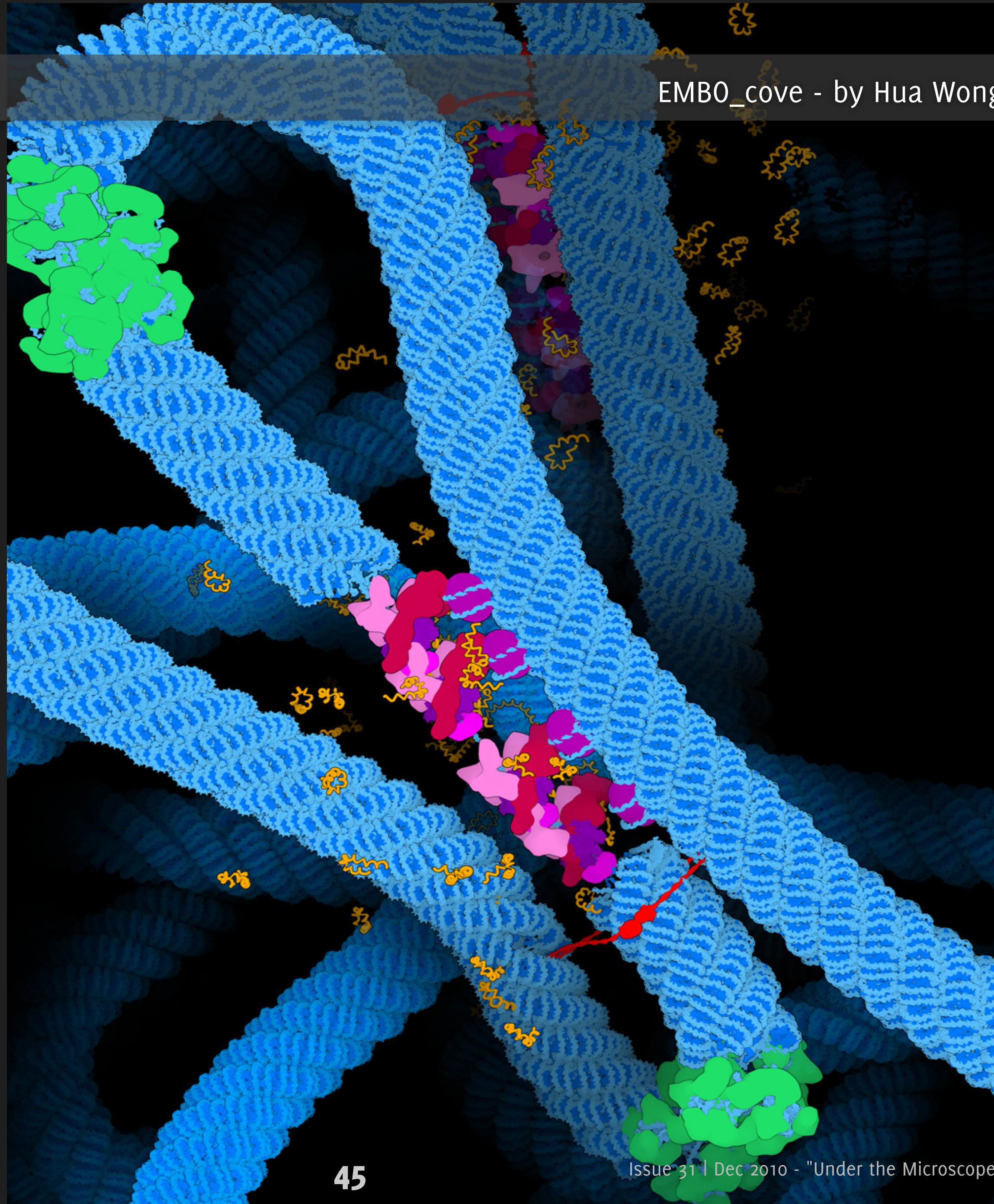


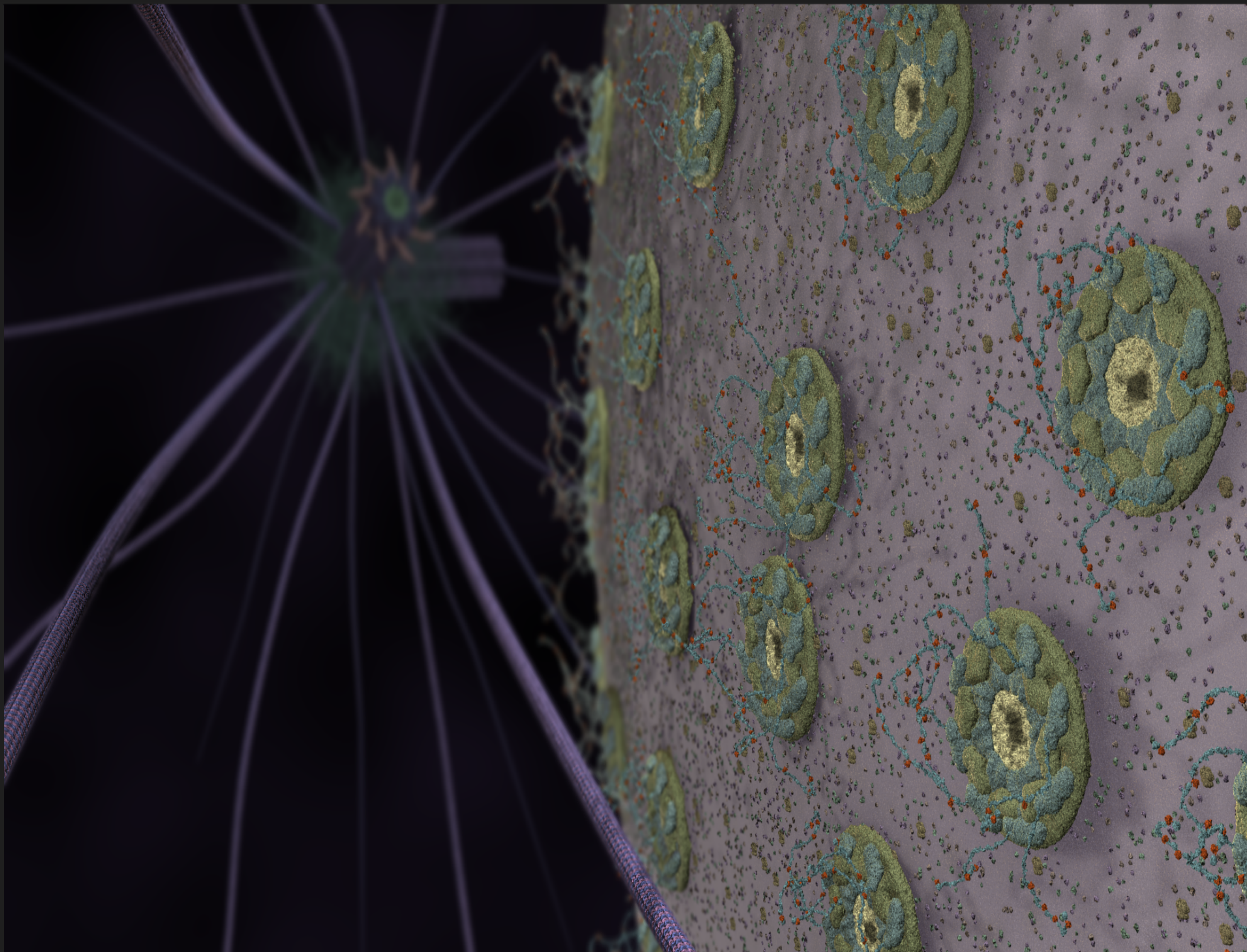

microlifeform 4 :: robert j. tiess :: © 2010

Quantum - by Sam Brubaker

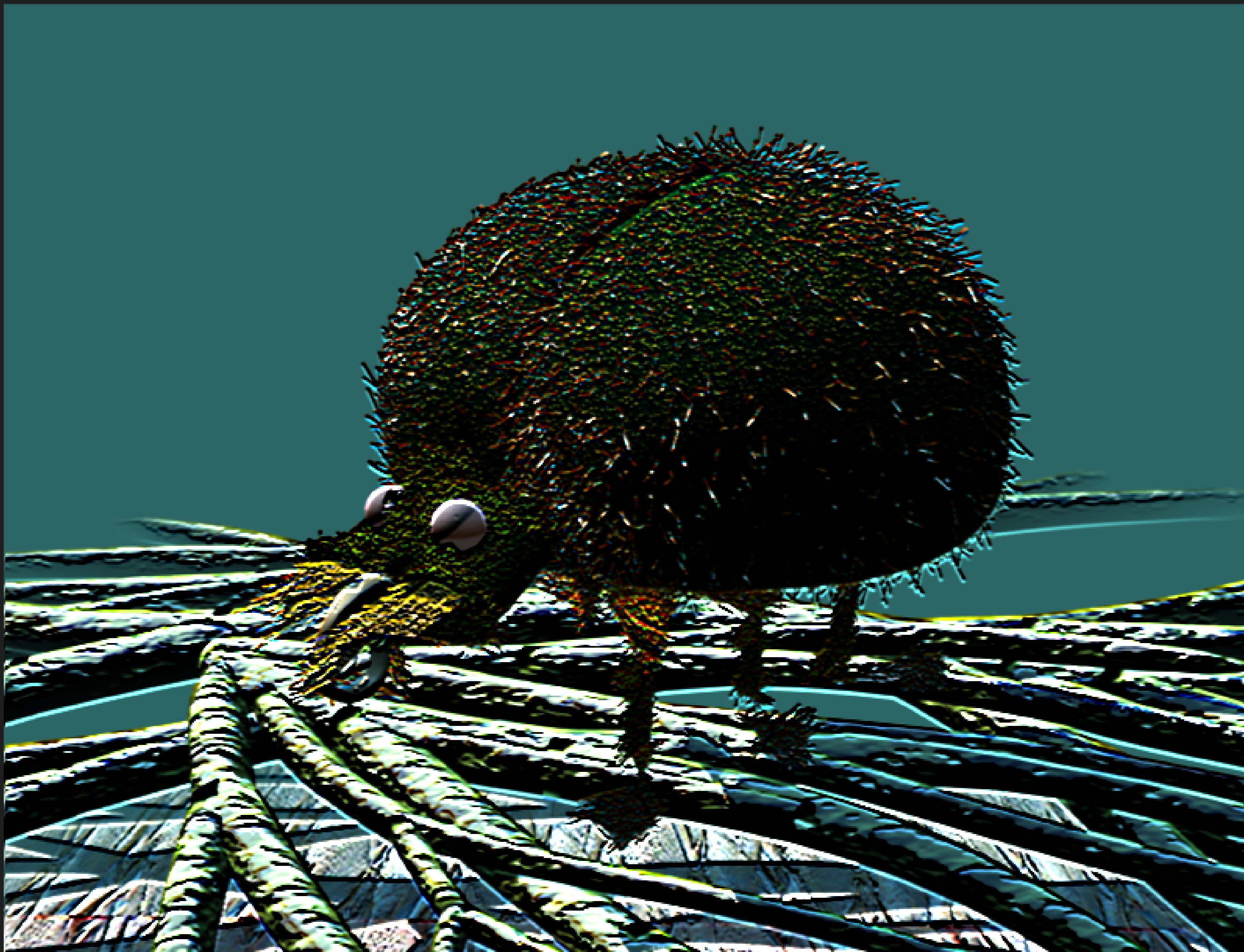


EMBO_cove - by Hua Wong

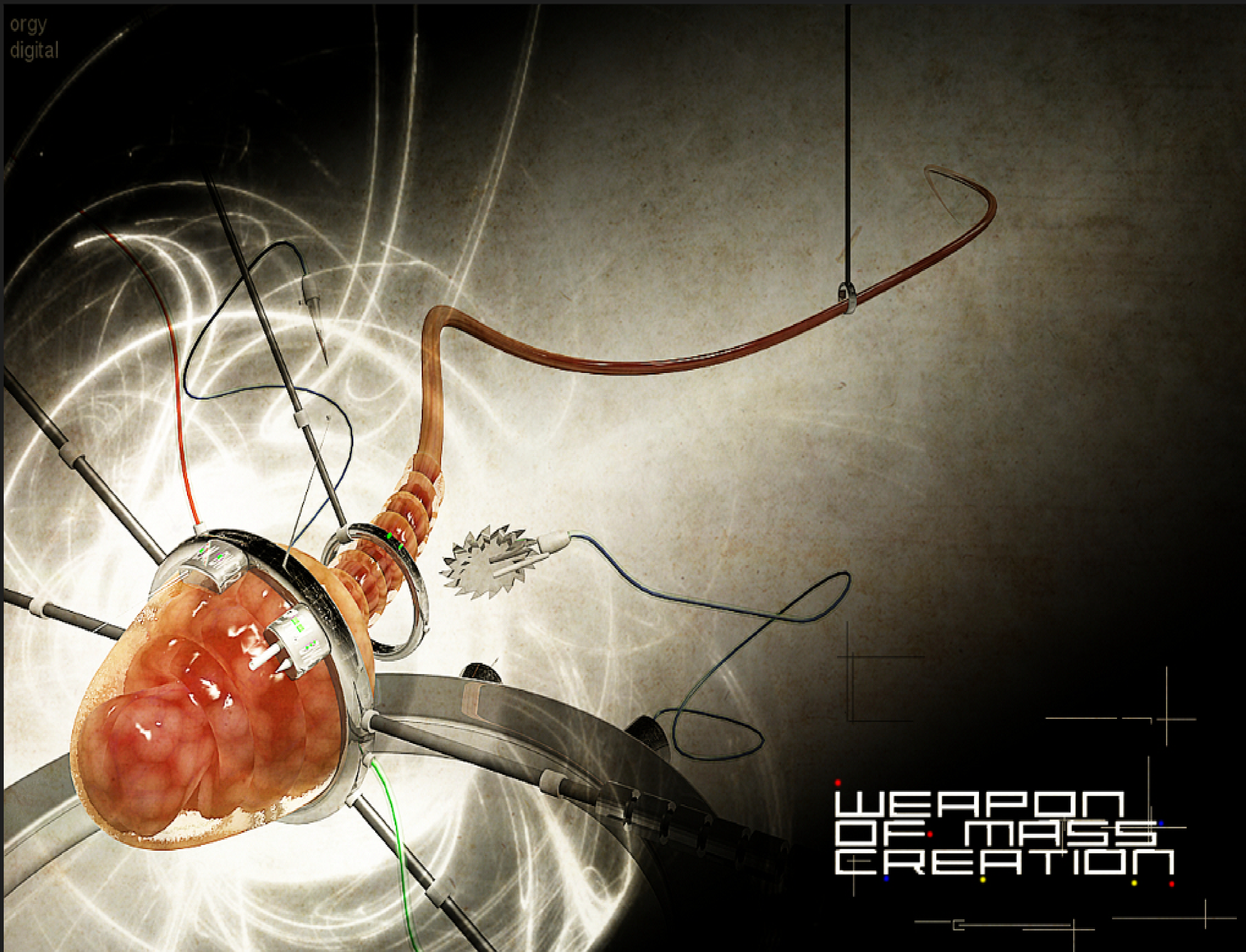


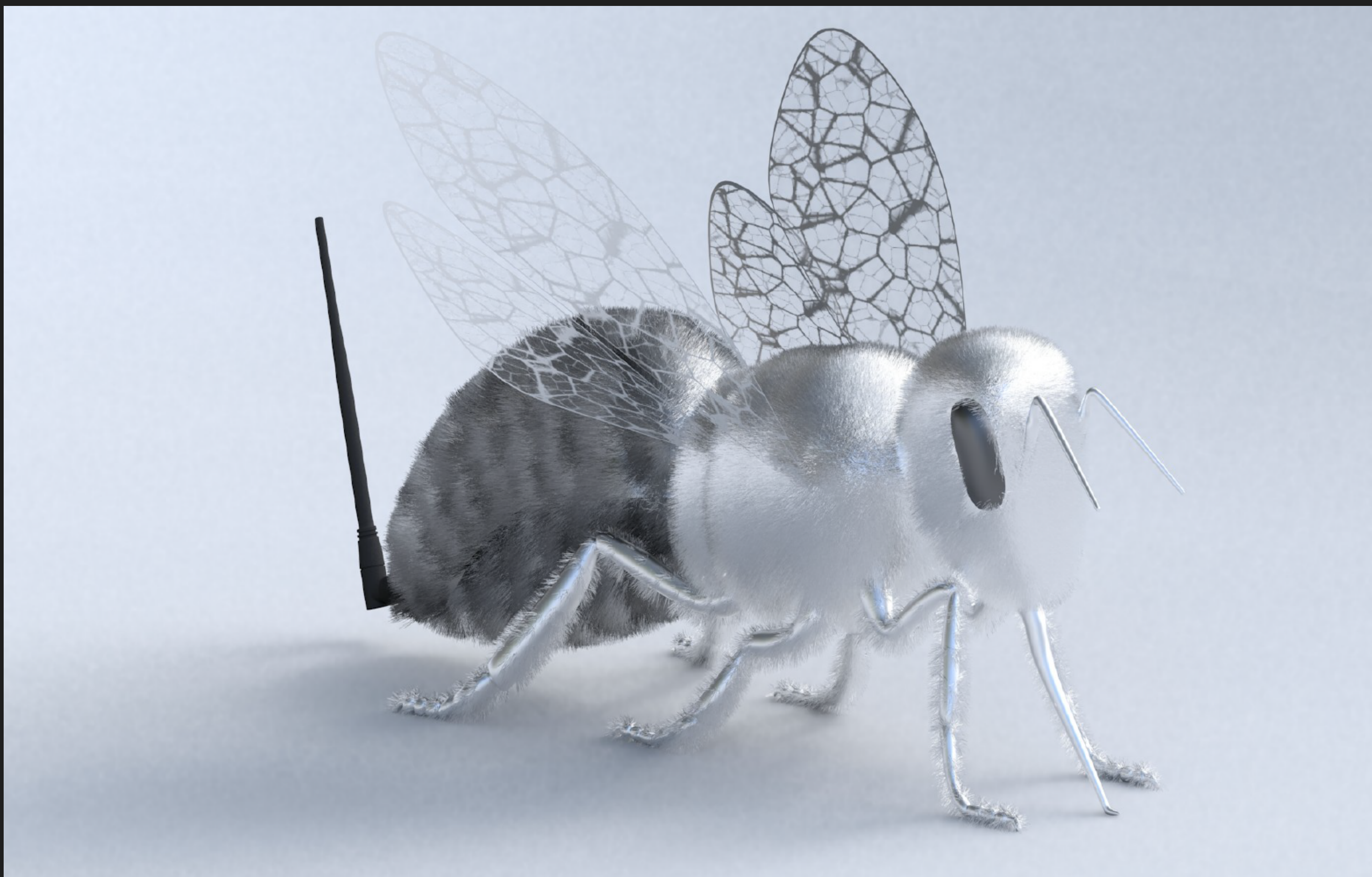


Cellulose Insect -by Antoni Villacreces



Weapon Of Mass Creation -by Yo Roque






50
Petronas
Malaysia
50
SYNTIUM
Blender 2.49 b
BMW z4 2006 By Pierlot Damien

Here is how!

1. We accept the following:

- Tutorials explaining new Blender features, 3dconcepts, techniques or articles based on current theme of the magazine.
- Reports on useful Blender events throughout the world.
- Cartoons related to blender world.

2. Send submissions to sandra@blenderart.org. Send us a notification on what you want to write and we can follow up from there. (Some guidelines you must follow)

- Images are preferred in PNG but good quality JPG can also do. Images should be separate from the text document.
- Make sure that screenshots are clear and readable and the renders should be at least 800px, but not more than 1600px at maximum.
- Sequential naming of images like, image 001.png... etc.
- Text should be in either ODT, DOC, TXT or HTML.
- Archive them using 7zip or RAR or less preferably zip.

3. Please include the following in your email:

- Name: This can be your full name or blenderartist avtar.
- Photograph: As PNG and maximum width of 256Px. (Only if submitting the article for the first time)
- About yourself: Max 25 words .
- Website: (optional)

Note: All the approved submissions can be placed in the final issue or subsequent issue if deemed fit. All submissions will be cropped/modified if necessary. For more details see the blenderart website.

Upcoming Issue 'Theme'

Issue 32

"Spring is Sprung"

- Modeling and texturing of plants, flowers, trees; can be realistic, toony, exotic, alien or even steampunk
- Ant Landscape Add-On
- Ivy Generator or similar scripts
- Use of arrays, curves and other modifiers to create vegetation

Disclaimer